

THÈSE  
présentée pour obtenir le grade de  
DOCTEUR DE L'ÉCOLE POLYTECHNIQUE

Spécialité :  
MATHÉMATIQUES – INFORMATIQUE

par  
JÉRÉMY BERTHOMIEU

**Contributions à la résolution des systèmes algébriques :  
réduction, localisation, traitement des singularités ;  
implantations**

Soutenue le 6 décembre 2011 devant le jury composé de :

M. KARIM BELABAS	Université de Bordeaux I	Examineur
M. VINCENT COSSART	Université de Versailles	Examineur
M. SHUHONG GAO	Clemson University	Rapporteur
M. MARC GIUSTI	CNRS & École polytechnique	Directeur
M. GRÉGOIRE LECERF	CNRS & École polytechnique	Codirecteur
M. BRUNO SALVY	INRIA Paris – Rocquencourt	Examineur
M. MICHAEL SHUB	University of Toronto	Rapporteur



# Remerciements

Je tiens à remercier tout d'abord Grégoire LECERF qui est, sans aucun doute, le grand instigateur de cette thèse. Il a su répondre avec une patience, une pédagogie et une disponibilité sans faille à toutes mes questions, mathématiques ou non, intelligentes ou non !

Je souhaite aussi remercier Marc GIUSTI pour avoir accepté d'être mon directeur de thèse et pour m'avoir toujours prodigué de sages conseils que ce soit pour les mathématiques ou pour les (trop) nombreuses tâches administratives auxquelles nous devons faire face.

Je souhaite exprimer ma gratitude à Luis Miguel PARDO pour m'avoir accueilli un trimestre dans son université à Santander et m'avoir fait découvrir une autre branche du calcul formel. Merci à Joris VAN DER HOEVEN pour sa disponibilité quant aux fréquentes questions à propos de  $\text{T}_{\text{E}}^{\text{X}}_{\text{MACS}}$  et de MATHEMAGIX, deux logiciels utilisés dans la préparation de ce manuscrit.

Je remercie vivement Shuhong GAO et Michael SHUB pour avoir accepté de rapporter cette thèse.

J'ai beaucoup apprécié apprendre et travailler avec Vincent COSSART, c'est pourquoi je le remercie de participer à mon jury de thèse. Je remercie aussi Karim BELABAS et Bruno SALVY pour avoir accepté de faire partie de mon jury de thèse.

Cette thèse n'aurait pu se faire sans la participation financière de l'École polytechnique *via* l'allocation internationale Gaspard MONGE, de l'Université de Versailles – St-Quentin-en-Yvelines *via* les postes de moniteur et d'attaché temporaire d'enseignement et de recherche que j'y ai occupés. Elle a aussi en partie été soutenue par le projet MAGIX ANR-09-JCJC-0098-01 et l'allocation DIGITEO 2009-36HD.

Plus haut, je parlais des nombreuses tâches administratives. Je remercie évidemment celles qui nous aident pour y faire face : Corinne POULAIN et Évelyne RAYSSAC à l'École polytechnique, Nadège ARNAUD, Laure FRÈREJEAN, Chantal POSTADJIAN et Liliane ROGER à l'Université de Versailles – St-Quentin-en-Yvelines.

Un grand merci aux membres des équipes MAX et TANC, en particulier Antoine COLIN, Daouda N. DIATTA, Francis JAMET et François OLLIVIER, du projet ALGO et, de manière générale, du GDT-CALCUL FORMEL avec qui j'ai pris grand plaisir à discuter. Je remercie aussi tous les membres des Laboratoire et Département de Mathématiques de Versailles auprès de qui j'ai appris, pendant cinq ans, puis discuté et enseigné, pendant trois ans, avec bonheur. Les nommer tous serait trop long mais je pense à Aurélie CORTEZ, Ariane MÉZARD, Christine POIRIER, Martin ANDLER, Alexis DEVULDER, Stéphane GINOULLAC, Otared KAVIAN, Mohamed KRIR, Yvan MARTEL, Guillermo MORENO-SOCÍAS, Luc ROBBIANO et Julien WORMS.

Trois ans de doctorat, c'est long et pour cela on a besoin d'amis doctorants. Je souhaite donc remercier tous ceux que j'ai pu côtoyer pendant ce doctorat ou un peu avant. Tout d'abord Romain LEBRETON grâce à qui, travailler pendant un tournoi du Grand Chelem n'est plus une torture ! Je pense aussi évidemment à mes autres *cobureaux* au LMV, d'hier et d'aujourd'hui : Jean-Maxime LABARBE et Sylvain ERVEDOZA pour leur accueil chaleureux et leurs âmes de grands frères, Pascal HIVERT pour son humour de folie, Cécile MAILLER, Aurélien GREUET et François DROUOT pour les très bons moments passés ensemble à papoter et enfin Claudio MUÑOZ, Reda SAHNOUN et Stefano MORRA pour leur soutien. Je n'oublie pas non plus Clémence DURVYE, Vanessa VITSE et les (anciens) thésards du bas comme Inès KAMOUN FATHALLAH, Éric FEKETE, Hamdi FATHALLAH, le capitaine Hussein MOURTADA pour son utilisation universelle de  $y^2 - x^3$ , Kowir PAMBO BELLO et Vianney COMBET. Merci aux doctorants et autres jeunes membres du LIX et de CAL4DOC avec qui j'ai eu le plaisir de discuter travail et crêpe : Cécile GONÇALVES, Sorina IONICA, Tania RICHMOND, Alain COUVREUR, Alexandre BENOIT, Guillaume QUINTIN, Jérôme MILAN, Luca DE FEO, Marc MEZZAROBBA, Morgan BARBIER, Olivier SCHWANDER et Pierre LAIREZ.

Évidemment, je n'oublie pas mes amis qui ont choisi une voie plus raisonnable. Nicolas pour me supporter en tant que colocataire. Binôme (ou Julie), qui connaît enfin le stress de la question fatidique « Alors la thèse, ça avance ? », Jérôme et Olivier pour nos super années passées à l'université. Un grand merci à Catherine et Alexandre pour le classique *resto-taru* ou *weekend-taru* mensuel ! Je remercie aussi ma marraine Brigitte et tous les gens de Dax pour leur accueil et les weekends à la française pendant mon séjour à Santander. Merci aussi à tous ceux que j'ai pu oublier mais qui ont toujours porté un intérêt certain à la préparation de cette thèse.

Mes derniers remerciements, mais non les moindres, sont bien sûr pour ma famille : mon frère Arnaud, Géraldine et ma nièce-filleule Juliette, ma sœur Chloé, Thibault et ma grand-mère. Un grand merci à mes parents pour nous avoir toujours soutenus, inconditionnellement, mon frère, ma sœur et moi, dans nos décisions quelles qu'elles fussent.

Enfin, merci à Camille pour sa gentillesse et son soutien dans la dernière ligne droite.

# Table des matières

<b>Remerciements</b> .....	3
<b>Index</b> .....	9
<b>Introduction</b> .....	11
1 Minimisation du nombre de variables additives .....	11
2 Algorithmes détendus pour les entiers $p$ -adiques .....	13
3 Résolution détendue des systèmes algébriques .....	16
4 Réduction de polynômes à deux variables .....	19
5 Résolution en moyenne des systèmes algébriques réels .....	21
Bibliographie .....	24
<b>Chapitre I Computing Hironaka’s invariants: Ridge and Directrix</b> .....	27
Abstract .....	27
1 Introduction .....	27
Acknowledgment .....	28
2 Notation and prerequisites, naive definitions of Ridge and Directrix .....	28
3 The Ridge: formal definition, main properties. ....	29
3.1 Ridge as a functor .....	29
3.2 Naive and formal definitions coincide .....	33
4 An algorithm to compute the ridge and the directrix .....	35
4.1 An algorithm to compute a “Giraud basis” of the cone .....	35
4.2 From the “Giraud basis” to the ridge .....	36
Bibliography .....	39
<b>Chapitre II Relaxed algorithms for <math>p</math>-adic numbers</b> .....	41
Abstract .....	41
1 Introduction .....	41
1.1 Motivation .....	42
1.1.1 Recursive equations .....	42
1.1.2 Elementary operations .....	43
1.1.3 User-friendly interface .....	43
1.2 Our contributions .....	43
Acknowledgments .....	45
2 Data structures and naive implementations .....	45
2.1 Finite $p$ -adic expansions .....	45
2.2 Classical complexities .....	45
2.3 The relaxed computational model .....	46
2.4 Addition .....	47
2.5 Subtraction in $\mathbb{Z}_p$ .....	47
2.6 Naive product .....	48
2.7 Lifting the power series product .....	49
2.8 Timings .....	50
3 Relaxed product .....	50
3.1 Fast relaxed multiplication algorithm .....	51
3.2 Variant with conversion to binary representation .....	54
3.3 Timings .....	56

4	Blockwise product	57
4.1	Relaxed versus zealous	57
4.2	Monoblock strategy	58
4.3	Relaxed blockwise product	59
4.4	Timings	60
5	Application to recursive $p$ -adic numbers	61
5.1	Implementation	61
5.2	Complexity analysis	62
5.3	Timings	63
6	Relaxed division	64
6.1	Division by a “scalar”	64
6.2	Quotient of two $p$ -adic numbers	65
6.3	Timings	65
7	Higher order roots	66
7.1	Regular case	66
7.2	$p$ th roots	68
7.3	Square roots in base 2	69
7.4	$p$ th roots in base $p$	69
7.5	Timings	71
8	Conclusion	71
	Bibliography	72
	<b>Chapitre III Relaxed <math>p</math>-adic Hensel lifting for algebraic systems</b>	<b>75</b>
	Abstract	75
1	Introduction	75
1.1	Preliminaries	75
1.2	Models of computation	76
1.3	Relaxed recursive $p$ -adic numbers	77
2	Simple root lifting of univariate polynomials	80
2.1	Dense polynomials	81
2.2	Polynomials as straight-line programs	82
3	Relaxed linear algebra over $p$ -adic numbers	84
3.1	Inversion of a “scalar” matrix	85
3.2	Inversion of a matrix over $R_p$	86
4	Root lifting for locally regular algebraic systems	87
4.1	Dense algebraic systems	87
4.2	Algebraic systems as straight-line programs	89
	Bibliography	90
	<b>Chapitre IV Reduction of bivariate polynomials from convex-dense to dense</b>	<b>91</b>
	Abstract	91
1	Introduction	91
1.1	Sizes of polynomials	91
1.2	Main result	92
1.3	Applications	93
Greatest common divisor	94	
Squarefree factorization	94	
Irreducible factorization	95	
1.4	Related work	95
	Acknowledgments	96
2	Support reduction	96
2.1	Bounding rectangles	96
2.2	Elementary transformations	97
2.3	Reduced sets of points	98
2.4	Degenerate case	99

2.5	Reduction algorithm	100
2.6	Bit-cost analysis	101
3	Dense size of reduced sets	102
3.1	Continuous bound	102
3.2	Discrete bound	107
4	Faster reduction algorithm	108
4.1	Dichotomic approach	108
4.2	Proof of Theorem IV.2	111
4.3	Timings	111
4.4	Optimality of the reduction	112
	Bibliography	113
<b>Chapitre V Spherical Radon transform and condition number</b>		117
	Abstract	117
1	Introduction	117
1.1	The context of our new results	117
1.2	Statement of the main results	120
1.3	The case of polynomial equations	123
	Acknowledgments	127
2	The underlying geometry	127
2.1	Some known facts about Grassmannian, Schubert and incidence varieties	127
2.2	The Schubert variety $\mathcal{L}_M$ : Proof of Lemma V.14	130
3	Some geometric integration tools	132
3.1	Normal Jacobians and the Co-area formula	135
3.2	Distances in $\mathcal{L}_M$ : Some technical results	136
3.3	Normal Jacobians I: Proof of Proposition V.21	137
3.4	Normal Jacobians II: Proof of Proposition V.22	140
3.5	Fibers over “complex” points: Proof of Proposition V.23	143
3.5.1	Proof of Proposition V.23	145
3.5.2	Proof of Remark V.24	147
4	Proof of the main results	148
4.1	Proof of Theorem V.5	148
4.2	Proof of Corollary V.7	151
4.3	Proof of Proposition V.8	152
5	Proof of the statements related to polynomial equation solving.	152
5.1	Proof of Corollary V.9	152
5.2	Proof of Corollaries V.10, V.11 and V.12	153
	Bibliography	155
<b>Annexe Introduction (translated into English)</b>		159
a	Minimization of the number of additive variables	159
b	Relaxed algorithms for $p$ -adic integers	161
c	Relaxed resolution of algebraic systems	164
d	Reduction of bivariate polynomials	167
e	Resolution in average of real algebraic systems	169
	Bibliography	171
<b>Annexe A Implementations in C++</b>		175
<b>Annexe B Examples in MATHEMAGIX</b>		179
a	The basics with $p$ -adic integers	179
b	Computation of $r$ th roots in $\mathbb{Z}_p$	181
b.a	Separable roots	181
b.b	$p$ th roots in $\mathbb{Z}_p$	182



# Index

A		
	Absolute factorization . . . . .	96
	Addition in $\mathbb{Z}_p$ . . . . .	47, 175
	Affine group $\text{Aff}(\mathbb{Z}^2)$ . . . . .	93, 112
	Algorithm	
	Computation of $\theta_i$ 's algorithm . . . . .	38
	Dense polynomial shifted algorithm . . . . .	82
	Dense polynomial system shifted algo. . . . .	88
	Dichotomic support reduction algorithm . . . . .	109
	Giraud basis algorithm . . . . .	35
	Ridge generators algorithm . . . . .	36
	Support reduction algorithm . . . . .	100
	Variant of BP algorithm for real systems . . . . .	124
	Appell function $F_1$ . . . . .	134, 148
	Approximate zero . . . . .	117, 120, 124
	Average complexity . . . . .	117
B		
	Bézout number . . . . .	123, 154
	Blockwise product . . . . .	57
	Bombieri's norm . . . . .	123
	Bounding octagon . . . . .	97
	Bounding rectangle . . . . .	91, 96
C		
	C++ . . . . .	44, 46, 175
	Co-area formula . . . . .	135
	Complexity	
	Average complexity . . . . .	117
	Integers multiplication complexity I . . . . .	45
	Multiplicative complexity $L^*$ . . . . .	76
	Polynomials multiplication complexity M . . . . .	45
	Relaxed multiplication complexity R . . . . .	43
	Computation of $\theta_i$ 's algorithm . . . . .	38
	Computation tree model . . . . .	93
	Condition number . . . . .	124
	Cone . . . . .	28
	Conversion to binary representation . . . . .	54, 175
	Convex size . . . . .	92
	Convex-dense . . . . .	92
D		
	Dense	
	algebraic systems . . . . .	87
	polynomial . . . . .	81
	polynomial shifted algorithm . . . . .	82
	polynomial system shifted algorithm . . . . .	88
	size . . . . .	91
	size of reduced sets . . . . .	102
	Dichotomic support reduction algorithm . . . . .	109
	Directrix . . . . .	28, 39
	Discriminant variety . . . . .	119, 152
E		
	Effective ring . . . . .	39, 41, 68, 75
F		
	Factorization	
	Absolute factorization . . . . .	96
	Irreducible factorization . . . . .	95
	Squarefree factorization . . . . .	94
	Faster reduction . . . . .	108
	Fewnomial systems . . . . .	118
G		
	Gaussian distribution . . . . .	125
	Geometric integration tools . . . . .	132
	Giraud basis . . . . .	31
	Giraud basis algorithm . . . . .	35
	Gröbner basis . . . . .	30
	Grassmannian and incidence varieties . . . . .	127
	Grassmannian of great circles on a sphere . . . . .	120
	Greatest common divisor (g.c.d.) . . . . .	94
H		
	Hasse-Schmidt derivation . . . . .	30, 36
	Hensel lifting . . . . .	43, 80, 95
	Homotopic deformation . . . . .	119
	$\eta$ -reduced set . . . . .	98
I		
	Incidence variety . . . . .	130
	Integers multiplication complexity I . . . . .	45
	Integral convex hull . . . . .	91, 107
	Irreducible factorization . . . . .	95
K		
	Kronecker substitution . . . . .	46, 57, 88, 95
L		
	Las Vegas algorithm . . . . .	118
	Lazy series . . . . .	41
M		
	MATHEMAGIX . . . . .	44, 82, 175, 179
	Maximal contact variety . . . . .	27
	Mignotte's bound . . . . .	43
	Models of computation . . . . .	76
	Multiplicative complexity $L^*$ . . . . .	76
N		
	next function . . . . .	46, 175
	Naive implementation . . . . .	45
	Naive product . . . . .	48
	Newton operator . . . . .	42, 62, 124
	Newton polygon . . . . .	92
	Normal Jacobian . . . . .	135
	Normalized set . . . . .	93
O		
	Optimality of the reduction . . . . .	112
P		
	$p$ -adic expansion . . . . .	45, 179
	$p$ -adic integer . . . . .	41, 175, 179
	Polynomials as s.l.p. . . . .	82
	Polynomials multiplication complexity M . . . . .	45
	Precision . . . . .	42, 175, 179
	Projective distance $d_{\mathbb{P}}$ . . . . .	120
	$p$ th root . . . . .	68, 182
Q		
	Questor set . . . . .	124, 153
R		
	Recursive equation . . . . .	42
	Relaxed	
	linear algebra over $p$ -adic numbers . . . . .	84
	multiplication complexity R . . . . .	43

- product . . . . . 51, 175
- quotient . . . . . 64, 177
- recursive  $p$ -adic number . . . . . 61, 77, 177
- series . . . . . 41
- Ridge . . . . . 28
- Ridge generators algorithm . . . . . 36
- Riemannian distance  $d_R$  . . . . . 120
- Root lifting
  - of multivariate algebraic systems . . . . . 87
  - of univariate polynomials . . . . . 80
- $r$ th root . . . . . 66, 181
- S
  - Scheme . . . . . 29
  - Shift . . . . . 78
  - Shifted algorithm . . . . . 80
  - Smale's 17th Problem . . . . . 117, 118
  - Smoothed analysis . . . . . 118
  - "Soft-Oh" notation  $\tilde{O}$  . . . . . 93
  - Sparse size . . . . . 92
  - Spherical Radon transform . . . . . 120
- Squarefree factorization . . . . . 94
- Stiefel manifold . . . . . 128
- Straight-line program (s.l.p.) . . . . . 76, 82, 89, 118
- Subtraction in  $\mathbb{Z}_p$  . . . . . 47, 175
- Support of a polynomial . . . . . 92
- Support reduction algorithm . . . . . 100
- T
  - Timings
    - of the blockwise product in  $\mathbb{Z}_p$  . . . . . 60
    - of the naive product in  $\mathbb{Z}_p$  . . . . . 50
    - of the relaxed product in  $\mathbb{Z}_p$  . . . . . 56, 57
    - of the relaxed quotient in  $\mathbb{Z}_p$  . . . . . 65
    - of the relaxed square root in  $\mathbb{Z}_p$  . . . . . 71
    - of the support reduction algorithm . . . . . 111
- U
  - Underlying geometry . . . . . 127
- V
  - Variant of BP algorithm for real systems . . . . . 124
- Z
  - Zealous series . . . . . 41

# Introduction

La résolution des systèmes algébriques est un problème au centre de l'algèbre et en particulier de la branche de la géométrie algébrique. Ses applications en mathématiques ou dans l'industrie sont nombreuses. Historiquement et techniquement, ce problème est indissociable du traitement des multiplicités. Cette thèse concerne certains aspects particuliers liés à l'efficacité d'une telle résolution. Tout d'abord, au chapitre I, nous nous intéressons à la minimisation du nombre de variables additives dans un tel système. Nous mentionnons des applications en cryptographie où des systèmes, *a priori* sous-déterminés, doivent être résolus en caractéristique positive. Puis, au chapitre II, nous proposons une arithmétique rapide pour calculer le produit, le quotient et plus généralement le point fixe d'une fonction définie sur  $\mathbb{Q}_p$ . Nous utilisons ensuite cette arithmétique au chapitre III afin de faire de la résolution locale dans  $\mathbb{Q}_p$  pour des systèmes s'évaluant bien. Au chapitre IV, nous considérons la décomposition des courbes planes en composantes irréductibles lorsque la taille du polygone de Newton est petite face au produit des degrés partiels. Enfin, au chapitre V, nous nous intéressons à la recherche de racines complexes approchées d'un système lorsque les coefficients sont dans  $\mathbb{R}$ .

*An English version of this introduction can be found in Appendix.*

## 1 Minimisation du nombre de variables additives

Le chapitre I est l'objet d'un travail en commun avec P. Hivert et H. Mourtada intitulé *Computing Hironaka's invariants: Ridge and Directrix* et publié dans *Arithmetic, Geometry, Cryptography and Coding Theory 2009* [BHM10].

Étant donné un idéal  $I$  engendré par des polynômes  $f_1, \dots, f_r \in \mathbb{K}[X_1, \dots, X_n]$ , on appelle *variété affine définie par  $I$* , l'ensemble  $V(I)$  des points de  $\bar{\mathbb{K}}^n$  qui annulent tous les polynômes de l'idéal  $I$ , où  $\bar{\mathbb{K}}$  est la clôture algébrique de  $\mathbb{K}$ . Un *point singulier* de la variété est un point  $x = (x_1, \dots, x_n) \in V(I)$  tel que pour tout  $f \in I$ , les dérivées partielles de  $f$  s'annulent aussi en  $x$ , c'est-à-dire tel que

$$f(x_1, \dots, x_n) = \frac{\partial f}{\partial X_1}(x_1, \dots, x_n) = \dots = \frac{\partial f}{\partial X_n}(x_1, \dots, x_n) = 0.$$

Dans le cas d'un polynôme  $f$  en deux variables  $X$  et  $Y$ ,  $V(f)$  est une *courbe plane*  $\mathcal{C}$ . Les singularités les plus simples que peut présenter  $\mathcal{C}$  sont des *points multiples*, ou auto-intersection, et des *points de rebroussement*, ou *cusps* en anglais, qui sont des points où la courbe n'admet que des demi-tangentes.

Par exemple, la courbe définie par  $f(X, Y) = Y^2 - X^3$  est singulière en l'origine, c'est un point de rebroussement. La *désingularisation* consiste à trouver une courbe non singulière  $\mathcal{C}'$  telle qu'il existe un morphisme birationnel de  $\mathcal{C}'$  vers  $\mathcal{C}$ .

Depuis les travaux de Hironaka dans les années 1960 et 1970, le problème de la désingularisation est bien compris en caractéristique nulle (voir [Hir64, Hir67, Hir70]). Le cas de la caractéristique positive est en revanche, lui, plus compliqué. Par exemple, la désingularisation de variétés de dimension 3 n'a été faite que récemment par Cossart et Piltant (voir [CP08, CP09]). Il faut cependant noter que leur preuve n'est pas constructive. Le cas de la dimension 4 est encore un problème ouvert. Dans [Hir64, Hir67], Hironaka introduit deux invariants pour la résolution de singularités : le *faîte* et la *directrice*. Étant donné un idéal homogène  $I \subseteq \mathbb{K}[X_1, \dots, X_n]$  et un cône  $\mathcal{C} = \text{Spec } \mathbb{K}[X_1, \dots, X_n]/I$ , ces deux invariants sont des sous-ensembles du cône tangent de la variété au voisinage d'un point  $x \in \mathcal{C}$ . En fait, d'après Giraud [Gir75], le faîte est le cône tangent d'une variété ayant un contact maximal avec  $\mathcal{C}$  au voisinage de  $x$ . La directrice est un espace vectoriel, c'est le plus grand sous-espace vectoriel  $W$  de  $\mathbb{A}^n = \text{Spec } \mathbb{K}[X_1, \dots, X_n]$  vérifiant  $\mathcal{C} + W = \mathcal{C}$ . Plus formellement, on a la définition suivante.

**Définition 1.** Soit  $\mathcal{C} = \text{Spec } \mathbb{K}[X_1, \dots, X_n]/I$  avec  $I \subseteq \mathbb{K}[X_1, \dots, X_n]$ , un idéal homogène. La directrice de  $\mathcal{C}$  est la plus grande famille libre  $(Y_1, \dots, Y_f)$ , où les  $Y_j$  sont des formes linéaires en les  $X_i$ , telle que

$$I = (I \cap \mathbb{K}[Y_1, \dots, Y_f]) \mathbb{K}[X_1, \dots, X_n].$$

Autrement dit, la directrice est le plus petit ensemble de variables nécessaires pour définir  $I$ .

Une définition analogue existe pour le faîte qui, lui, est juste un sous-groupe additif de  $\mathbb{K}^n$ .

**Définition 2.** Le faîte de  $\mathcal{C}$  est le plus grand sous-groupe additif de  $\mathbb{A}^n$  engendré par le plus petit ensemble de polynômes additifs  $P_1, \dots, P_e$  tel que

$$I = (I \cap \mathbb{K}[P_1, \dots, P_e]) \mathbb{K}[X_1, \dots, X_n].$$

Puisque les notions de forme linéaire et de polynôme additif coïncident en caractéristique nulle, il est clair que le faîte et la directrice sont confondus dans ce cas. Ceci a pour conséquence qu'une variété de contact maximal est toujours lisse. Cependant, ce n'est plus vrai en caractéristique positive  $p$  et alors, une variété de contact maximal peut ne pas être lisse. Cet obstacle empêche de généraliser la démonstration de la résolution de singularité de Hironaka à la caractéristique positive. Nous nous intéressons à ces deux invariants de variété que sont le faîte et la directrice. Nous y suivons les travaux de Giraud [Gir72, Gir75] à ce sujet qui introduisent une définition fonctorielle du faîte.

**Proposition-Définition 3.** Soit  $\mathbb{A}_{\mathbb{K}}^n$  l'espace affine sur  $\mathbb{K}$  de dimension  $n$ . Soit  $F$  le foncteur associant à un  $\mathbb{K}$ -schéma  $S$  l'ensemble  $F(S)$  des points  $v \in \mathbb{A}_{\mathbb{K}}^n$  tels que pour tout  $S$ -point  $c \in \mathcal{C}(S)$ ,

$$(v + c) \in \mathcal{C}(S).$$

Alors le foncteur  $F$  est représentable par un schéma  $F$  que nous appelons le *faîte* de  $\mathcal{C}$ .

Nous obtenons le résultat suivant.

**Corollaire 4. (Corollary I.14, [BHM10, Corollary 2.12])** Soit  $U$  l'algèbre des fonctions définies sur  $\mathbb{A}_{\mathbb{K}}^n$  telles que pour tout  $\mathbb{K}$ -schéma  $S$  et tout  $S$ -point  $(u, v)$  de  $F \times_{\mathbb{K}} \mathbb{A}_{\mathbb{K}}^n$ , on a  $f(u + v) = f(u)$ .

Soit  $I$  un idéal homogène de  $\mathbb{K}[X_1, \dots, X_n]$  et soit  $\mathcal{G} := \{\gamma_1, \dots, \gamma_s\}$  une base de Gröbner réduite homogène de  $J$  l'idéal du faîte de  $V(I)$ , alors

$$I = (I \cap \mathbb{K}[\gamma_1, \dots, \gamma_s]) \mathbb{K}[X_1, \dots, X_n].$$

De plus,  $U = \mathbb{K}[\gamma_1, \dots, \gamma_s]$  et si  $K$  est une  $\mathbb{K}$ -algèbre engendrée par des polynômes additifs telle que

$$I = (I \cap K) \mathbb{K}[X_1, \dots, X_n],$$

alors  $U \subset K$ .

Par conséquent, les définitions 2 et 3 du faîte coïncident.

Nous donnons, de plus, un algorithme permettant de calculer le faîte d'un idéal. Nous montrons aussi que si  $(P_1, \dots, P_e)$  est une famille génératrice du faîte, alors il existe des entiers  $\alpha_1, \dots, \alpha_e$  tels que la famille  $(\sqrt[p^{\alpha_1}]{P_1}, \dots, \sqrt[p^{\alpha_e}]{P_e})$  est génératrice de la directrice. Dans le cas d'un corps parfait  $\mathbb{K}$ , nous pouvons alors en déduire un algorithme de calcul de la directrice. Malheureusement, savoir si un élément est une puissance  $p^\alpha$ -ième peut se révéler indécidable en caractéristique  $p$ , et ce, même si l'anneau considéré est effectif (cf. [FS56]).

## 2 Algorithmes détendus pour les entiers $p$ -adiques

Le chapitre II est une adaptation avec J. van der Hoeven et G. Lecerf de l'algorithme détendue des séries formelles aux entiers  $p$ -adiques. Ce travail a fait l'objet d'un article *Relaxed algorithms for  $p$ -adic numbers* accepté pour publication au *Journal de Théorie des Nombres de Bordeaux* [BHL11].

La *normalisation* consiste à résoudre les singularités de codimension 1. Dans le cas d'une courbe singulière  $\mathcal{C}$ , les lieux singuliers sont des points, ils sont donc tous de codimension 1. Ainsi, il suffit de normaliser  $\mathcal{C}$  pour trouver  $\mathcal{C}'$ , la désingularisée de  $\mathcal{C}$ . Si la caractéristique de  $\mathbb{K}$  est 0, alors normaliser  $\mathcal{C} = V(f)$  est équivalent à calculer la *fermeture intégrale* de  $\mathbb{K}[X]$  dans  $\mathbb{K}(X)[Y]/(f)$  (cf. [Sha94, Chapter II, Section 5]). Soient  $R$  et  $R'$  deux anneaux tels que  $R$  est un sous-anneau de  $R'$ , on dit que  $b \in R'$  est *entier* sur  $R$  si, de manière équivalente [Lan02, Chapter VII], on a

- le sous-anneau  $R[b]$  est de type fini sur  $R$  ;

- il existe  $m \in \mathbb{N}^*$  et  $a_0, \dots, a_{m-1} \in R$  tel que  $b$  est racine du polynôme  $T^m + a_{m-1}T^{m-1} + \dots + a_0$ .

La fermeture intégrale  $\bar{R}$  de  $R$  dans  $R'$  est alors l'ensemble des  $b \in R'$  entiers sur  $R$ . La notion d'être entier étant stable par les opérations d'anneaux,  $\bar{R}$  est à la fois un sous-anneau de  $R'$  et un  $R$ -module.

**Exemple 5.** En reprenant l'exemple précédent,  $f(X, Y) = Y^2 - X^3$ , il est clair que  $X^{3/2} = Y \in \mathbb{K}(X)[Y]/(f)$  est entier sur  $\mathbb{K}[X]$  et donc que la fermeture intégrale  $\overline{\mathbb{K}[X]}$  de  $\mathbb{K}[X]$  contient  $\mathbb{K}[X, X^{3/2}]$ . Cependant, on peut remarquer que  $X^{1/2} = \frac{Y}{X}$  est racine de  $T^2 - X$ , par conséquent  $\mathbb{K}[X, X^{1/2}] = \mathbb{K}[X^{1/2}] \subseteq \overline{\mathbb{K}[X]}$ . Il s'avère que cette inclusion large est, en fait, une égalité.

Dans le cas où  $R$  est principal, la fermeture intégrale  $\bar{R}$  de  $R$  dans une extension finie de  $K = \text{Frac } R$  est un  $R$ -module libre de type fini et de rang  $n$ . Au lieu de calculer une base de  $\bar{R}$ , il est en général plus facile de calculer une base de son complété  $\bar{R}_p$  pour la valuation  $p$ -adique dans un des localisés  $K_p$  de  $K$ , où  $p$  est un premier de  $R$ . Citons [Hal01, Hoe94, Tra84] qui proposent de tels algorithmes. Il suffit de faire ces calculs de bases *locales* pour  $p$  tel que  $p^2 \mid \text{disc } f$ . L'avantage du calcul de ces bases locales est que chaque élément  $b_i$  de la base  $(b_0, \dots, b_{n-1})$  s'écrit sous forme

$$b_i = \frac{b_{i,0} + \dots + b_{i,n-1} \alpha^{n-1}}{p^{d_i}},$$

où pour tout  $j$ ,  $0 \leq j \leq n-1$ ,  $b_{i,j} \in R_p$  et pour tout  $i$ ,  $\text{val}_p(b_i) \geq 0$ . De plus, nous connaissons une borne pour l'exposant  $d_i$  présent au dénominateur :

$$2 d_i \leq \text{val}_p(\text{disc } f).$$

En fait, on peut améliorer cette majoration et l'on obtient [Hoe94, Section 2.3]

$$2(d_0 + \dots + d_{n-1}) \leq \text{val}_p(\text{disc } f). \quad (1)$$

Ensuite, il faut recoller les différentes bases locales en une base *globale*.

**Exemple 6.** Le polynôme  $f(Y) = Y^2 - 5$  définit l'extension  $\mathbb{Q}(\sqrt{5})$  sur  $\mathbb{Q}$ . Puisque  $\sqrt{5}$  est entier sur  $\mathbb{Z}$ , on sait que la fermeture intégrale de  $\mathbb{Z}$  contient  $\mathbb{Z}[\sqrt{5}]$ . Comme le discriminant de  $f$  est  $-4 = -2^2$ , le calcul d'une seule base locale, celle dans  $\mathbb{Q}_2(\sqrt{5})$ , suffit. D'après l'équation (1), une base  $(1, b_1)$  peut au mieux vérifier que  $b_1 = \frac{1+b_{1,1}\sqrt{5}}{2}$  avec  $b_{1,1} \in \mathbb{Z}_2$ . Sur  $\mathbb{F}_2$ , le polynôme  $f$  se factorise en

$$f(Y) = Y^2 + 1 = (Y + 1)^2,$$

par conséquent, le développement diadique de  $\sqrt{5}$  commence par 1 et donc  $\text{val}_2\left(\frac{1+\sqrt{5}}{2}\right) = 0$ . On en déduit que  $b_1 = \frac{1+\sqrt{5}}{2}$  est entier. Il s'agit en fait du nombre d'or  $\varphi$ , racine de  $T^2 - T - 1$ . On n'a qu'une seule base locale, donc il n'y a pas de problème de recollement et la fermeture intégrale de  $\mathbb{Z}$  dans  $\mathbb{Q}(\sqrt{5})$  est  $\mathbb{Z}[\varphi]$ .

Le calcul d'une base intégrale en dimension 1, c'est-à-dire de l'anneau des entiers sur  $\mathbb{K}[X]$  dans un *corps de fonctions*  $\mathbb{K}(X)[Y]/(f)$  est très similaire à celui d'une base intégrale de l'anneau des entiers dans un *corps de nombres*  $\mathbb{Q}[Y]/(f)$ . Nous avons vu plus haut que les algorithmes les plus courants calculaient d'abord des bases dans les localisés pour espérer, ensuite, obtenir une base globale. Pour un corps de fonctions, ces localisés sont les corps des séries de Laurent  $\mathbb{K}((X - \alpha))$ , avec  $\alpha \in \mathbb{K}$  si  $\mathbb{K} = \bar{\mathbb{K}}$ , pour un corps de nombres, ce sont les corps des nombres  $p$ -adiques  $\mathbb{Q}_p$ , avec  $p \in \mathbb{N}$  premier. Pour implanter une arithmétique sur de tels objets, principalement deux représentations s'offrent à l'utilisateur. La première, dite *zélée*, consiste à travailler à précision fixe  $n$ , c'est-à-dire qu'une série formelle  $S \in \mathbb{K}[[X]]$ ,

$$S = \sum_{k=0}^{\infty} S_k X^k,$$

sera représentée sous la forme d'un polynôme tronqué de degré  $n$  :

$$S = S_0 + \dots + S_{n-1} X^{n-1} + O(X^n).$$

Cette représentation propose l'avantage d'utiliser toutes les routines rapides de multiplications de polynômes, en particulier la multiplication de deux séries en précision  $n$  se calcule en  $\mathbf{M}(n) \in \tilde{O}(n)$  opérations dans le corps de base  $\mathbb{K}$ . En contrepartie, si au cours des calculs, une perte de précision a lieu au point que le résultat final ne convienne pas, il est nécessaire de reprendre les calculs depuis le début en augmentant la précision souhaitée.

Une seconde façon de représenter les séries calculables est de les voir comme un vecteur de coefficients calculés  $(S_0, \dots, S_\ell)$  muni d'une fonction `next( $n$ )` capable de renvoyer le coefficient  $S_n$  à partir des précédents. Par exemple, l'exponentielle calculée jusqu'à la précision 10 sera représentée par le vecteur  $(1, \dots, \frac{1}{9!})$  et par la fonction `next( $n$ )` renvoyant  $S_0 = 1$  si  $n = 0$  et  $S_n := \frac{S_{n-1}}{n}$  sinon. Une telle représentation est appelée  *paresseuse*  car les coefficients d'une série ne sont calculés que lorsque le besoin est réel. Il semble naturel de définir la méthode `next( $n$ )` du produit  $ST$  par le renvoi naïf de  $S_0 T_n + S_1 T_{n-1} + \dots + S_n T_0$ , malheureusement, ceci nous donne une complexité quadratique en la précision pour la multiplication des séries :  $\mathbf{M}(n) \in O(n^2)$ . En revanche, la perte de précision n'est plus un problème, si l'utilisateur a besoin des quelques coefficients suivants, il lui suffit de faire appel à la fonction `next` autant de fois que nécessaire. Et ce, sans reprendre les calculs depuis le début. En 1997, puis en 2002, van der Hoeven a proposé une multiplication rapide de ces séries paresseuses qu'il nomma *détendue* [Hoe97, Hoe02]. L'idée est de s'autoriser à faire plus de calculs que nécessaire à certains moments, en faisant des produits de polynômes de degrés 1, 2, 4, ..., afin de profiter de la complexité de tels produits (voir figure II.1 au chapitre II). Comparé au produit zélé, le surcoût est au pire logarithmique de sorte que le produit de deux séries détendues se calcule jusqu'à la précision  $n$  en  $\mathbf{R}(n) \in O(\mathbf{M}(n) \log n)$  opérations dans  $\mathbb{K}$ .

Nous présentons une adaptation à tous les complétés  $I$ -adiques d'un anneau. La motivation principale est la gestion de la retenue qui peut apparaître en effectuant une addition ou une multiplication. En général, ces anneaux contiennent  $\mathbb{Z}_p$  comme sous-anneau, c'est pourquoi nous nous sommes en particuliers intéressés aux anneaux d'entiers  $p$ -adiques. Cependant, ce ne sont pas les seuls.

**Exemple 7.** Complétons l'anneau  $\mathbb{R}[X]$  pour la valuation  $(X^2 + 1)$ -adique. L'idéal  $(X^2 + 1)$  étant premier, le complété  $\mathbb{R}[X]_{(X^2+1)}$  est intègre et est l'ensemble des éléments  $S$  de la forme

$$S = \sum_{k=0}^{\infty} (a_k X + b_k) (X^2 + 1)^k.$$

L'addition de deux séries  $S$  et  $T$  se fait bien composante par composante, en revanche la multiplication peut générer une retenue. Si  $S = 2X - 1$ , alors

$$S^2 = (2X - 1)^2 = 4X^2 - 4X + 1 = -(4X + 3) + 4(X^2 + 1).$$

Notons cependant que comme le corps résiduel de  $\mathbb{R}[X]_{(X^2+1)}$  est  $\mathbb{R}[X]/(X^2 + 1) \simeq \mathbb{C}$ , alors, d'après [Coh46, Theorem 15], cet anneau est isomorphe à  $\mathbb{C}[[T]]$  qui, lui, ne demande aucune gestion de retenue. Un de ces isomorphismes envoie  $T$  sur  $X^2 + 1$  et  $i$  sur  $X + \frac{1}{2}X(X^2 + 1) + \frac{3}{8}X(X^2 + 1)^2 + \dots$ .

Nous montrons entre autres que les complexités obtenues pour les séries formelles se transposent aux entiers  $p$ -adiques. Notons  $l(m)$  le coût du produit de deux entiers dont la taille ne dépasse pas  $m$  bits et  $l_p(n)$  le coût du produit de deux développements  $p$ -adiques d'ordre  $n$  et dont les coefficients ont une représentation binaire usuelle.

**Proposition 8. (Propositions II.6 et II.7, [BHL11, Propositions 3.1 et 3.2])**  
Soient  $a$  et  $b$  deux entiers  $p$ -adiques détendus. Le produit  $a b$  peut être calculé jusqu'à la précision  $n$  en utilisant  $O(l_p(n) \log n)$  opérations sur les bits. Pour ce calcul, l'espace mémoire total nécessaire pour stocker les différentes retenues ne dépasse pas  $O(n)$ .

À l'aide de conversions de la base 2 vers la base  $p$  et vice versa, le calcul de  $a b$  jusqu'en précision  $n$  peut être fait en  $O(l(n \log p) \log n)$  opérations sur les bits et peut nécessiter  $O(n \log p)$  bits d'espace mémoire.

Une série récursive est une série formelle  $S$  point fixe d'une fonction  $\Phi$ , c'est-à-dire telle que  $S = \Phi(S)$ . Si de plus, le calcul du  $n$ -ième terme  $\Phi(S)_n$  ne requiert que la connaissance de  $S_0, \dots, S_{n-1}$ , pour tout  $n \geq k$ , alors  $S$  est d'ordre  $k$  et à partir des  $k$  premiers termes de  $S$  et de la fonction  $\Phi$ , on peut calculer récursivement n'importe quel terme de  $S$ . Si une série est inversible, alors son inverse est une série récursive d'ordre 1. Nous adaptons aussi les algorithmes pour les  $p$ -adiques récursifs, en particulier, celui du calcul de l'inverse. Nous en concluons que les complexités de la proposition 8 sont encore valables pour le calcul du quotient  $a/b$  de deux entiers  $p$ -adiques en précision  $n$ . En section 7, nous donnons une méthode récursive de calcul de la racine  $r$ -ième dans  $\mathbb{Z}_p$  d'un entier  $p$ -adique. Si  $r$  est premier avec  $p$ , le lemme de Hensel nous assure qu'une telle racine existe dès lors qu'il en existe une modulo  $p$ .

### 3 Résolution détendue des systèmes algébriques

Le chapitre III constitue un travail en cours avec R. Lebreton intitulé *Relaxed  $p$ -adic Hensel lifting for algebraic systems* [BL12] généralisant le calcul récursif d'une racine  $r$ -ième dans  $\mathbb{Z}_p$ .

Considérons les systèmes à coefficients rationnels. Savoir si de tels systèmes ont des solutions rationnelles peut être compliqué : les tailles des numérateurs et des dénominateurs des solutions peuvent être très grandes. Même en utilisant une arithmétique rapide sur  $\mathbb{Q}$ , comme celle disponible avec GMP [G+91], un surcoût non négligeable provient des réductions en fractions irréductibles. On peut préférer résoudre ces systèmes dans  $\mathbb{Q}_p$ , le complété  $p$ -adique de  $\mathbb{Q}$ , pour un ou plusieurs  $p$  bien choisis. Parmi toutes les solutions trouvées, il faut alors déterminer lesquelles sont susceptibles d'être aussi dans  $\mathbb{Q}$ .

**Exemple 9.** Prenons le cas le plus simple d'un système réduit à un polynôme à une variable :

$$f(X) = X^4 - 1.$$

Clairement, ce polynôme n'admet que deux racines rationnelles 1 et -1. D'ailleurs, si  $p=2$  ou si  $p=3 \pmod{4}$ , alors il se factorise sur  $\mathbb{Q}_p$  sous la forme attendue

$$f(X) = (X - 1)(X + 1)(X^2 + 1).$$

Pourtant, pour certains  $p$  premiers, en fait ceux congrus à 1 modulo 4, on trouve 4 racines dans  $\mathbb{Q}_p$ . Par exemple, si  $p=5$ ,  $f(X)$  se factorise en

$$\begin{aligned} f(X) = & (X - 1)(X - (4 + 4p + 4p^2 + 4p^3 + O(p^4))) \\ & \times (X - (2 + p + 2p^2 + p^3 + O(p^4)))(X - (3 + 3p + 2p^2 + 3p^3 + O(p^4))). \end{aligned}$$

Il faut savoir alors détecter, à partir de ces 4 racines, celles qui sont éventuellement rationnelles. Ici, grâce aux développements périodiques des deux premières racines, il est clair que ce sont celles dans  $\mathbb{Q}$ . On peut noter que les deux autres racines sont alors tout simplement des représentants de  $i$  et de  $-i$  dans  $\mathbb{Q}_5$ .

En général, la période d'un rationnel peut être suffisamment grande pour ne pas être détectée. On a donc besoin de savoir reconstruire un rationnel à partir d'un développement  $p$ -adique. La borne de Mignotte [GG03, Chapter 6] est une borne sur la taille des coefficients des facteurs d'un tel polynôme  $f$ . En particulier, ceci nous fournit une borne sur la précision nécessaire avec laquelle on doit factoriser  $f$  dans  $\mathbb{Q}_p$  pour en trouver les racines dans  $\mathbb{Q}$ .

Dans le cas plus général d'un système à coefficients entiers dont on connaît une solution  $x = (x_1, \dots, x_r) \in \mathbb{Q}_p^r$ , on peut utiliser la *reconstruction rationnelle* (cf. [GG03, Section 5.10]) sur chaque  $x_i$  afin de voir si  $x_i \in \mathbb{Q}$ . Si c'est le cas,  $x \in \mathbb{Q}^r$  est une solution rationnelle du système. Des bornes sur les précisions nécessaires pour faire une telle reconstruction existent, elles sont liées à ce qui est appelé le *Bézout arithmétique* (voir [BGS94, KPS01, McK01]).

La proposition suivante nous donne la complexité de calcul d'un vecteur d'entiers  $p$ -adiques récursifs.

**Proposition 10.** Soit  $\Phi$  une expression contenant  $L$  instructions de type addition, soustraction ou produit. Soit  $\mathbf{y} \in \mathbb{Z}_p^r$  récursif d'ordre  $k$  tel que  $\mathbf{y} = \Phi(\mathbf{y})$  et  $\mathbf{y}_0, \dots, \mathbf{y}_{k-1}$  sont connus, alors le calcul de  $\mathbf{y}_n$  peut se faire en  $O(L \log_p(n) \log n)$  opérations en utilisant le produit détendu de la proposition 8.

Parmi les différents algorithmes de résolution des systèmes algébriques, KRONECKER, présenté dans [GLS01, DL08], résout des systèmes représentés en évaluation, par exemple représentés par un *straight-line program (s.l.p.)*. Il faut, pour cela, que ceux-ci n'admettent qu'un nombre fini de solutions dans la clôture algébrique  $\overline{\mathbb{K}}$  de  $\mathbb{K}$ , on dit aussi qu'ils doivent être *zéro-dimensionnels*. L'intérêt d'une telle représentation est que certains polynômes peuvent avoir beaucoup de monômes tout en étant très facilement évaluables. On peut citer par exemple  $(X_1 + \dots + X_r)^d$  qui s'évalue en  $O(r + \log d)$  multiplications grâce à l'*exponentiation rapide*, alors que, développé, il possède  $\binom{r+d-1}{d}$  monômes. Ou le déterminant d'une matrice carrée de taille  $r$  qui est un polynôme en  $r^2$  variables avec  $r!$  termes mais qui s'évalue en  $O(r^3)$  opérations grâce au pivot de Gauss.

Au chapitre III, nous donnons une méthode automatique permettant de calculer récursivement une racine  $y \in \mathbb{Z}_p$  d'un polynôme  $f(Y)$  telle que  $y_0 = y \bmod p$  soit racine simple de  $f(Y) \bmod p$ . Tout d'abord, soit  $\Phi$  une fonction pour laquelle  $y$  est un point fixe si, et seulement si,  $y$  est un zéro de  $f$ . En pratique nous prenons toujours

$$\Phi(Y) = \frac{f'(y_0)Y - f(Y)}{f'(y_0)}.$$

Il n'est pas clair qu'une telle fonction définisse un entier  $p$ -adique récursif. Nous montrons que nous pouvons toujours trouver une fonction  $\Psi$  faisant de  $y$  un entier  $p$ -adique récursif d'ordre 1 et dont la complexité d'évaluation est linéaire en celle de  $f$ . Plus précisément nous avons la proposition suivante.

**Proposition 11. (Proposition III.18, [BL12, Proposition 18])** *Soit  $f$  un polynôme à une variable sur  $\mathbb{Z}_p$  donné sous forme de straight-line program tel que sa complexité multiplicative soit  $L^*$ . Alors il existe une fonction  $\Psi$  obtenue à partir de  $\Phi$  et de  $y_0$  telle que*

$$y = \Psi(y)$$

*et pour tout  $n \in \mathbb{N}^*$ , le calcul de  $\Psi(y)_n$  ne demande que la connaissance de  $y_0, \dots, y_{n-1}$ . De plus, la complexité d'évaluation de  $\Psi$  est majorée par  $2L^* + 1$ .*

De même, nous montrons que le calcul du quotient de deux entiers  $p$ -adiques peut se généraliser en le calcul de la solution d'un système linéaire régulier. Nous finissons par traiter le cas d'un système algébrique  $\mathbf{f}$  de  $r$  équations polynomiales à  $r$  variables à coefficients dans  $\mathbb{Z}$ . Nous supposons que la réduction de  $\mathbf{f}$  modulo  $p$ ,  $\mathbf{f}_0$ , admet une racine régulière  $\mathbf{y}_0$ , un vecteur d'entiers  $p$ -adiques. C'est-à-dire que nous supposons que la différentielle  $d\mathbf{f}_{\mathbf{y}_0}$  en  $\mathbf{y}_0$  est inversible sur  $\mathbb{F}_p$ . Cette différentielle est alors inversible sur  $\mathbb{Z}_p$ , le lemme de Hensel s'applique encore et les propositions 10 et 12 se généralisent à ce contexte.

**Proposition 12. (Proposition III.31, [BL12, Proposition 31])** *Soit  $\mathbf{f}$  un système polynomial à  $r$  variables sur  $\mathbb{Z}_p$  donné sous forme de straight-line program tel que sa complexité multiplicative soit  $L^*$ . Alors il existe une fonction  $\Psi$  obtenue à partir de  $\Phi$  et de  $\mathbf{y}_0$  telle que*

$$\mathbf{y} = \Psi(\mathbf{y})$$

et pour tout  $n \in \mathbb{N}^*$ , le calcul de  $\Psi(\mathbf{y})_n$  ne demande que la connaissance de  $\mathbf{y}_0, \dots, \mathbf{y}_{n-1}$ . De plus, la complexité d'évaluation de  $\Psi$  est majorée par  $3L^* + \frac{r(r+1)}{2}$ .

En particulier, nous pouvons espérer en déduire une amélioration de la complexité de l'algorithme KRONECKER de Giusti, Lecerf et Salvy (voir [GLS01, DL08]).

Nous effectuons aussi une étude de complexité dans le cas où  $\mathbf{f}$  est représenté de manière dense, c'est-à-dire où chaque polynôme de  $\mathbf{f}$  est donné par son vecteur de coefficients.

Tous les algorithmes présentés aux chapitres II et III ont été implantés en C++ dans le paquet ALGEBRAMIX du logiciel de calcul formel MATHEMAGIX [H+02]. Des exemples de code C++ de ces implantations sont disponibles en annexe A. Des exemples de leur utilisation pour le calcul du produit, du quotient et des racines  $r$ -ièmes et  $p$ -ièmes avec MATHEMAGIX sont eux en annexe B.

## 4 Réduction de polynômes à deux variables

Le chapitre IV est un travail en commun avec G. Lecerf intitulé *Reduction of bivariate polynomials from convex-dense to dense, with application to factorizations* et accepté pour publication à *Mathematics of Computation* [BL10]. Nous y étudions la factorisation d'un polynôme à deux variables. Plus particulièrement, nous cherchons à réduire un polynôme  $f \in \mathbb{K}[X, Y]$  en un polynôme  $\tilde{f}$  dont nous pouvons calculer la factorisation plus facilement afin d'en déduire celle de  $f$ . En effet, la factorisation de polynômes à deux variables est l'une des briques de base pour la décomposition en composantes irréductibles des hypersurfaces, c'est-à-dire des variétés définies par une seule équation. En effet, étant donné un polynôme à deux variables  $f(X, Y) \in \mathbb{K}[X, Y]$ , si l'on connaît une factorisation  $f(0, Y) = \tilde{f}_1(Y) \cdots \tilde{f}_s(Y)$  avec  $\tilde{f}_1, \dots, \tilde{f}_s$  tous premiers entre eux, alors par le lemme de Hensel, il existe  $f_1(X, Y), \dots, f_s(X, Y) \in \mathbb{K}[[X]][Y]$  tels que

$$f(X, Y) = f_1(X, Y) \cdots f_s(X, Y).$$

À partir d'un facteur  $f_i(X, Y) = \sum_{j=0}^{d_i} f_{i,j}(X) Y^j$ , il faut alors déterminer si les séries formelles  $f_{i,j}(X) \in \mathbb{K}[[X]]$  appartiennent à  $\mathbb{K}(X)$ . On peut utiliser pour cela les *approximants de Padé-Hermite* [GG03, Section 5.9]. On en conclut alors que  $f$  se factorise dans  $\mathbb{K}(X)[Y]$  et donc dans  $\mathbb{K}[X, Y]$ .

En répétant ce procédé, on peut en déduire un algorithme factorisant  $f(X_1, \dots, X_r)$  dans  $\mathbb{K}[X_1, \dots, X_r]$  dès lors que  $f(0, \dots, 0, X_r)$  admet des facteurs premiers entre eux dans  $\mathbb{K}[X_r]$ .

Si l'on suppose que  $f$  s'écrit sous la forme

$$f(X, Y) = \sum_{(i,j) \in \mathbb{N}^2} f_{i,j} X^i Y^j,$$

on appelle *support* de  $f$ , l'ensemble  $\mathcal{S} = \{(i, j) \in \mathbb{N}^2, f_{i,j} \neq 0\}$ . Le *polygone de Newton*, noté  $\mathcal{N}$ , est l'enveloppe convexe de  $\mathcal{S}$ . Il permet entre autres, grâce à l'algorithme de Newton-Puiseux, de calculer les racines de  $f$  lorsque celui-ci est vu comme un polynôme en  $Y$  à coefficients dans  $\mathbb{K}[X]$  (voir [Wal78, Chapter IV] et [Die68, Chapitre III, Appendice]). Il peut servir aussi de critère d'irréductibilité de  $f$ , en effet si  $f = gh$ , alors  $\mathcal{N} = \mathcal{N}(g) + \mathcal{N}(h)$  [Ost21, Ost75, Ost99], où  $\mathcal{N}(g)$  et  $\mathcal{N}(h)$  sont les polygones de Newton de  $g$  et  $h$  respectivement et  $+$  représente la somme de Minkowski. Ce critère peut être vu comme une généralisation du critère d'Eisenstein à deux variables.

Notons  $\pi$  le nombre de points à coordonnées entières dans  $\mathcal{N}$ , il est appelé la *taille convexe* de  $\mathcal{N}$ .

Il est attendu que la complexité d'un algorithme de factorisation dépende de  $\pi$  ou de l'aire de  $\mathcal{N}$ , que nous notons  $\text{Vol } \mathcal{N}$ . Par exemple, dans [Wei10], sous certaines conditions de généricité, on peut trouver un algorithme pour factoriser  $f$  dont la complexité est en  $O(\pi^\omega)$ , où  $\omega$  est l'exposant de l'algèbre linéaire. Pourtant, les meilleures bornes de complexité connues pour les *factorisations sans carré et irréductibles* dépendent du produit des degrés partiels  $d_X$  en  $X$  et  $d_Y$  en  $Y$  (voir par exemple [Gao03, Lec07, Lec08, Lec10]). Notons, qu'*a contrario*, il n'est pas raisonnable d'espérer que la complexité d'un algorithme de factorisation dépende polynomialement du cardinal  $\sigma$  de  $\mathcal{S}$ . En effet, le polynôme  $f(X, Y) = X^p - Y^p \in \mathbb{Q}[X, Y]$ , avec  $p$  premier, est tel que  $\sigma = 2$  mais se factorise en

$$f(X, Y) = (X - Y)(X^{p-1} + X^{p-2}Y + \dots + Y^{p-1}),$$

le second facteur ayant un support de taille  $p$ .

Supposons que  $d_X \geq d_Y$ . En caractéristique zéro, la factorisation sans carré de  $f$  peut se calculer, de façon déterministe [Lec08, Proposition 8] (resp. probabiliste [Lec08, Proposition 9]), en  $\tilde{O}(d_X d_Y^2)$  (resp.  $\tilde{O}(d_X d_Y)$ ) opérations dans  $\mathbb{K}$ . En caractéristique positive  $p$ , la factorisation irréductible de  $f$  sur  $\mathbb{K} = \mathbb{F}_{p^k}$  peut se calculer, de façon probabiliste [Lec10], en  $\tilde{O}(k(d_X d_Y)^{1,5})$  opérations dans  $\mathbb{F}_p$ .

Ces complexités peuvent alors être très pessimistes. Par exemple, la famille de polynômes  $f_n(X, Y) = 1 + Y + X^n Y^n$  est telle que le produit des degrés partiels est  $n^2$ , pourtant, l'aire du polygone de Newton  $\mathcal{N}_n$  est  $n/2$ .

Le *rectangle englobant*  $\mathcal{R}$  de  $\mathcal{N}$  est le plus petit rectangle de la forme  $(o_X, o_Y) + [0, d_X] \times [0, d_Y]$  contenant  $\mathcal{N}$ . Le nombre de points à coordonnées entières dans  $\mathcal{R}$ , c'est-à-dire  $\delta = (d_X + 1)(d_Y + 1)$ , est la *taille dense* de  $\mathcal{N}$ .

**Théorème 13.** (Theorems IV.2 et IV.20, [BL10, Theorems 1.2 et 4.3])  
Soit  $\mathcal{S}$  un sous-ensemble fini de  $\mathbb{Z}^2$  de taille  $\sigma$ , d'enveloppe convexe  $\mathcal{N}$ , de taille convexe  $\pi$  et de taille dense  $\delta$ . Soit  $\mathcal{R} = (o_X, o_Y) + [0, d_X] \times [0, d_Y]$  le rectangle englobant de  $\mathcal{S}$ . Il existe une application affine inversible  $U \in \text{Aff}(\mathbb{Z}^2)$  telle que  $\tilde{\mathcal{N}} = U(\mathcal{N})$  est inclus dans un rectangle  $\tilde{\mathcal{R}} = [0, \tilde{d}_X] \times [0, \tilde{d}_Y]$  vérifiant

$$\frac{3}{8} \tilde{d}_X \tilde{d}_Y \leq \text{Vol } \tilde{\mathcal{N}} \leq \tilde{d}_X \tilde{d}_Y. \quad (2)$$

De plus,  $U$  se calcule en  $O(\sigma \log^2 \pi)$  opérations sur les bits et la taille dense de  $\tilde{\mathcal{N}}$  est au plus  $9\pi$ .

L'application  $U$  se calcule comme composée de *transvections*, de symétries axiales et de translations. D'un point de vue géométrique sur la courbe définie par  $f$ ,  $U$  est la composée d'éclatements et de contractions, de sorte que la factorisation de  $\tilde{f} = U(f)$  devient essentiellement équivalente à celle de  $f$ .

D'après l'équation (2), on peut remarquer que  $\text{Vol } \tilde{\mathcal{N}} \in O(\tilde{d}_X \tilde{d}_Y)$ . Or comme  $U$  est inversible sur  $\mathbb{Z}$ , alors  $\text{Vol } \tilde{\mathcal{N}} = \text{Vol } \mathcal{N}$  et  $\pi \in O(\tilde{d}_X \tilde{d}_Y)$ . On en déduit qu'en réduisant  $f$  à  $\tilde{f}$ , en calculant la factorisation souhaitée de  $\tilde{f}$  et en appliquant  $U^{-1}$  à ses facteurs, on peut retrouver la factorisation sans carré de  $f$ , de façon déterministe (resp. probabiliste), en  $\tilde{O}(\pi^{1,5})$  (resp.  $\tilde{O}(\pi)$ ) opérations dans  $\mathbb{K}$  lorsque  $\text{car } \mathbb{K} = 0$  et la factorisation irréductible, de façon probabiliste, de  $f$  en  $\tilde{O}(k \pi^{1,5})$  opérations dans  $\mathbb{F}_p$  lorsque  $\mathbb{K} = \mathbb{F}_{p^k}$ .

Enfin, nous montrons aussi que le facteur  $3/8$  apparaissant dans l'équation (2) est optimal, en général.

**Proposition 14. (Proposition IV.21, [BL10, Proposition 4.4])** *Soit  $\mathcal{S}$  un sous-ensemble de  $\mathbb{Z}^2$ . Avec la convention que  $\frac{\text{Vol } U(\mathcal{S})}{\text{Vol } \mathcal{R}(U(\mathcal{S}))} = 1$  dès que  $\text{Vol } \mathcal{S} = 0$ , on a*

$$\inf_{\mathcal{S} \subset \mathbb{Z}^2, |\mathcal{S}| < \infty} \sup_{U \in \text{Aff}(\mathbb{Z}^2)} \frac{\text{Vol } U(\mathcal{S})}{\text{Vol } \mathcal{R}(U(\mathcal{S}))} = \frac{3}{8},$$

où  $\mathcal{R}(U(\mathcal{S}))$  représente le rectangle englobant de  $U(\mathcal{S})$ .

## 5 Résolution en moyenne des systèmes algébriques réels

Dans le chapitre V, nous nous intéressons avec L. M. Pardo au calcul en moyenne d'une racine approchée d'un système algébrique réel. Ce chapitre est issu d'un article nommé *Spherical Radon transform and the average of the condition number on certain Schubert subvarieties of a Grassmannian* accepté pour publication à *Journal of Complexity* [BP11b].

Dans [SS93a, SS93b, BCSS98], les auteurs définissent un zéro approché  $z'$  d'un vrai zéro  $z$  d'un système algébrique complexe homogène  $f = (f_1, \dots, f_n) \in \mathbb{C}[X_0, \dots, X_n]^n$ , comme étant un élément de  $\mathbb{P}^n(\mathbb{C})$  pour lequel l'itération de l'opérateur de Newton converge immédiatement quadratiquement vers  $z$ . Soient  $d_R$  la distance de Riemann, naturelle, sur la sphère-unité de  $\mathbb{C}^{n+1}$  et  $d_{\mathbb{P}}$  la distance projective sur  $\mathbb{P}^n(\mathbb{C})$  définie par  $d_{\mathbb{P}}(w, w') = \sin(d_R(w, w'))$ . Plus formellement, si l'on note  $z_0 = z'$  et pour tout  $k \in \mathbb{N}$ ,  $z_{k+1} = z_k - d f_z |_{T_{f(z)}}^{-1}(f(z))$ , alors pour tout  $k$ , on a

$$d_{\mathbb{P}}(z, z_k) \leq \left(\frac{1}{2}\right)^{2^k - 1} d_{\mathbb{P}}(z, z').$$

Le dix-septième problème de Smale pose la question de déterminer si, étant donné un système homogène générique  $f$  de  $n$  polynômes en  $n + 1$  variables, il existe un algorithme capable de calculer un zéro approché  $z'$  de  $f$  en un temps polynomial. L'algorithme présenté par Beltrán et Pardo [BP09a, BP09b, BP11a] en est un exemple, sa complexité est en  $O(N^2)$ , où  $N$  est la taille dense de  $f$ . Cependant, cet algorithme est probabiliste. Le meilleur algorithme déterministe connu à ce jour est une adaptation du leur grâce à la *smooth analysis*, l'exposant apparaissant dans sa complexité est alors en  $O(\log \log N)$ .

Notons  $d_i = \deg f_i$  et  $(d) = (d_1, \dots, d_n)$ . Notons aussi  $\mathcal{H}_{(d)}$  l'espace des systèmes algébriques homogènes complexes  $g = (g_1, \dots, g_n)$  en les variables  $X_0, \dots, X_n$  tels que  $\deg g_i = d_i$ . Cet espace est muni naturellement de la norme de Bombieri et l'on peut dès lors supposer que tous les systèmes intervenant dans la suite sont de norme 1. En particulier on suppose que  $f$  l'est. Beltrán et Pardo proposent d'appliquer de la déformation homotopique le long d'un grand cercle de  $\mathbb{S}(\mathcal{H}_{(d)})$ , la sphère-unité de  $\mathcal{H}_{(d)}$ . En partant d'un système  $g$  dont ils connaissent une racine  $\zeta$ , ils étudient comment  $\zeta$  évolue le long du grand cercle joignant  $g$  à  $f$ . Pour que l'algorithme de Beltrán et Pardo rende une racine approchée de  $f$ , il ne faut pas que ce grand cercle coupe  $\Sigma$ , la variété discriminante. Cette variété  $\Sigma$  est la variété des systèmes de  $\mathbb{S}(\mathcal{H}_{(d)})$  dits *mal-conditionnés*, c'est-à-dire de ceux dont l'ensemble des zéros n'est pas de dimension zéro ou de ceux dont au moins un zéro est multiple. C'est une variété de codimension complexe 1 et donc de codimension réelle 2.

Les auteurs introduisent l'ensemble *questeur*  $\mathcal{G}_{(d)}$  qui est un ensemble de systèmes  $g$  dont ils savent calculer une racine exacte  $\zeta$ . En choisissant un couple  $(g, \zeta)$  au hasard dans  $\mathcal{G}_{(d)}$ , avec probabilité 1, le grand cercle passant par  $g$  et  $f$  ne coupe pas  $\Sigma$ . Par conséquent, avec probabilité 1, l'algorithme de Beltrán et Pardo renvoie une racine approchée de  $f$ .

Dans le chapitre V, nous nous intéressons au cas analogue où  $f$  est supposé à coefficients réels, ce que nous dénoterons par  $f \in \mathcal{H}_{(d)}^{\mathbb{R}}$ . L'adaptation directe de l'algorithme de Beltrán et Pardo à ce cas-ci n'est pas possible. En effet  $\Sigma^{\mathbb{R}} = \Sigma \cap \mathbb{S}(\mathcal{H}_{(d)}^{\mathbb{R}})$ , la variété discriminante réelle, sépare la sphère-unité  $\mathbb{S}(\mathcal{H}_{(d)}^{\mathbb{R}})$  en un nombre exponentiel de composantes connexes. Pour que la déformation homotopique puisse réussir, il faut alors espérer choisir  $g$  dans la même composante que  $f$ , ce qui n'est pas raisonnable. C'est pourquoi nous proposons de choisir  $g$  parmi les systèmes complexes au lieu des systèmes réels et d'y appliquer ensuite leur algorithme. Nous en déduisons que notre variante de l'algorithme de Beltrán et Pardo retourne une racine approchée de  $f$  avec probabilité 1.

Notons  $\mu_{\text{norm}}(g, \zeta)$  le conditionnement normalisé de  $g$  en sa racine  $\zeta$ . Ce nombre permet de déterminer à quel point une petite perturbation de  $g$  affecte  $\zeta$ . Il est lié au conditionnement de la jacobienne de  $g$  en  $\zeta$  et donc, en particulier, lié à la norme d'opérateur de l'inverse de  $d g_{\zeta}$  qui peut être notée  $\|d g_{\zeta}^{-1}\|$ . Le nombre d'étapes de déformation homotopique exécutée par l'opérateur de Newton à partir d'un couple  $(g, \zeta)$  en ciblant  $f$  le long du grand cercle  $L$  passant par  $g$  et  $f$  est majoré par

$$\mathcal{C}(f, g, \zeta) = \int_{h \in L} \mu_{\text{norm}}(h, \xi)^2 dL,$$

où  $\xi$  est le zéro de  $h$  issu de la perturbation de  $\zeta$  (voir [Shu09]). Une majoration de la complexité de notre algorithme est donnée par l'espérance  $E$  définie par

$$E = E_{\mathbb{S}(\mathcal{H}_{(d)}^{\mathbb{R}}) \times \mathbb{S}(\mathcal{G}_{(d)})}(\mathcal{C}(f, g, \zeta)).$$

Pour le calcul de cette espérance, on peut utiliser la moyenne  $\mu_{\text{av}}^2$  des carrés des conditionnements définie par :

$$\mu_{\text{av}}^2(g) = \frac{1}{|\mathbb{V}(g)|} \sum_{\zeta \in \mathbb{V}(g)} \mu_{\text{norm}}(g, \zeta)^2.$$

Notons  $N + 1 = \dim_{\mathbb{C}} \mathcal{H}_{(d)} = \dim_{\mathbb{R}} \mathcal{H}_{(d)}^{\mathbb{R}}$ , ainsi,  $N$  est la taille dense d'un système de  $\mathbb{S}(\mathcal{H}_{(d)}^{\mathbb{R}})$ . À l'aide de la transformée de Radon sphérique [Rub02] définie par

$$\mathbf{R}^{\alpha}[\mu_{\text{av}}^2](\mathbb{S}(\mathcal{H}_{(d)}^{\mathbb{R}})) = \frac{\mathbb{B}\left(\frac{N-\alpha+2}{2}, \frac{\alpha+N+1}{2}\right)}{\text{vol } \mathbb{S}(\mathcal{H}_{(d)})} \int_{\mathbb{S}(\mathcal{H}_{(d)})} \frac{\mu_{\text{av}}^2(g)}{\mathbb{d}_{\mathbb{P}}(g, \mathbb{S}(\mathcal{H}_{(d)}^{\mathbb{R}}))^{N+1-\alpha}} \mathrm{d} \mathbb{S}(\mathcal{H}_{(d)}),$$

nous en déduisons le résultat suivant.

**Théorème 15. (Corollary V.13, [BP11b, Corollary 12])** *Supposons que  $\dim_{\mathbb{R}} \mathcal{H}_{(d)}^{\mathbb{R}} = N + 1$  et notons*

$$C(2N + 1, N, i) = 2 \binom{N-1}{i} \frac{\mathbb{B}\left(\frac{2N+3}{2}, \frac{1}{2}\right)}{\mathbb{B}\left(\frac{N-1}{2}, \frac{1}{2}\right)}.$$

Alors la borne  $E$  de la complexité de la variante réelle de l'algorithme de Beltrán et Pardo vérifie,

1. si  $N + 1$  est pair,

$$E = \sum_{i=0}^{\frac{N-1}{2}} C(2N + 1, N, i) \mathbf{R}^{N-2i}[\mu_{\text{av}}^2](\mathbb{S}(\mathcal{H}_{(d)}^{\mathbb{R}})) ;$$

2. si  $N + 1$  est impair,

$$\begin{aligned} \mathbf{R}^1[\mu_{\text{av}}^2](\mathbb{S}(\mathcal{H}_{(d)}^{\mathbb{R}})) &\leq E \leq \sqrt{2} \left( \frac{\mathbf{R}^1[\mu_{\text{av}}^2](\mathbb{S}(\mathcal{H}_{(d)}^{\mathbb{R}}))}{\mathbb{B}\left(\frac{N+1}{2}, \frac{N}{2}\right)} \right) \\ E &\leq \sqrt{2} E_{\mathbb{S}(\mathcal{H}_{(d)})} \left[ \frac{\mu_{\text{av}}^2(g)}{\mathbb{d}_{\mathbb{P}}(g, \mathbb{S}(\mathcal{H}_{(d)}^{\mathbb{R}}))^N} \right]. \end{aligned}$$

Nos résultats sont en fait un peu plus généraux. Étant donné un  $\mathbb{R}$ -sous-espace vectoriel  $M$  de  $\mathcal{H}_{(d)}$ , nous calculons l'espérance  $E = E_{\mathbb{S}(M) \times \mathbb{S}(\mathcal{G}_{(d)})}(\mathcal{C}(f, g, \zeta))$  de trouver une racine approchée de  $f \in M$  de norme 1 à partir d'un couple  $(g, \zeta)$  de  $\mathcal{G}_{(d)}$ . Plus haut, l'espace  $M$  était celui des systèmes réels  $\mathcal{H}_{(d)}^{\mathbb{R}}$ , mais il peut très bien s'agir des systèmes algébriques à support inclus dans un ensemble  $\mathcal{S}$  donné. Le théorème 15 est alors un cas particulier du théorème suivant.

**Théorème 16. (Corollary V.10, [BP11b, Corollary 9])** Soient  $N + 1 = \dim_{\mathbb{C}} \mathcal{H}_{(d)}$  et  $p + 1 = \dim_{\mathbb{R}} M$ . Pour tout  $i$ , on note

$$B_0(2N + 1, p, i) = 2 \binom{N - 1 - \frac{p}{2}}{2} \frac{B(N + \frac{3}{2}, \frac{1}{2})}{B(\frac{p-1}{2}, \frac{1}{2})},$$

$$C(2N + 1, p, i) = 2 \binom{N - \frac{p+1}{2}}{i} \frac{B(N + \frac{3}{2}, \frac{1}{2})}{B(\frac{p-1}{2}, \frac{1}{2})}.$$

Soit  $E$  la borne de la complexité de la variante de l'algorithme de Beltrán et Pardo,

1. si  $\text{codim}_{\mathbb{R}} M = 1$ , alors

$$\frac{4\sqrt{2\pi}}{(2N + 1 + \sqrt{3})^{1/2}} E_{\mathbb{S}(\mathcal{H}_{(d)})}[\mu_{\text{av}}^2] \leq E \leq \sqrt{\frac{(2N - 1)\pi}{2}} \mathbf{R}^0[\mu_{\text{av}}^2](\mathbb{S}(M)) ;$$

2. si  $\text{codim}_{\mathbb{R}} M$  est paire, alors

$$E = \sum_{i=0}^{\frac{2N-p-1}{2}} C(2N + 1, p, i) \mathbf{R}^{2(N-i)-p}[\mu_{\text{av}}^2](\mathbb{S}(M)) ;$$

3. si  $\text{codim}_{\mathbb{R}} M$  est impaire et est plus grande que 1, alors  $E$  est bornée par les quantités suivantes :

$$E \geq \sum_{i=0}^{\frac{2N-p-2}{2}} \frac{(2N - 1) B_0(2N + 1, p, i)}{\sqrt{i + \sqrt{3/2}}} \mathbf{R}^{2(N-i)-p-1}[\mu_{\text{av}}^2](\mathbb{S}(M)),$$

$$E \leq \sum_{i=0}^{\frac{2N-p-2}{2}} \frac{8 B_0(2N + 1, p, i)}{(2i + 1)(2N - 1)} \mathbf{R}^{2(N-i)-p-1}[\mu_{\text{av}}^2](\mathbb{S}(M)).$$

## Bibliographie

- [BCSS98] L. Blum, F. Cucker, M. Shub et S. Smale. *Complexity and real computation*. Springer-Verlag, New York, 1998. With a foreword by Richard M. Karp.
- [BGS94] J.-B. Bost, H. Gillet et C. Soulé. Heights of projective varieties and positive Green forms. *J. Amer. Math. Soc.*, 7(4):903–1027, 1994.
- [BHL11] J. Berthomieu, J. van der Hoeven et G. Lecerf. Relaxed algorithms for  $p$ -adic numbers. *J. Théor. Nombres Bordeaux*, 23(3):541–577, 2011.
- [BHM10] J. Berthomieu, P. Hivert et H. Mourtada. Computing Hironaka's invariants: Ridge and Directrix. Dans *Arithmetic, Geometry, Cryptography and Coding Theory 2009*, volume 521 de *Contemp. Math.*, pages 9–20. Amer. Math. Soc., Providence, RI, 2010.
- [BL10] J. Berthomieu et G. Lecerf. Reduction of bivariate polynomials from convex-dense to dense, with application to factorization. Manuscript to appear in *Math. Comp.*, 2010.
- [BL12] J. Berthomieu et R. Lebreton. Relaxed  $p$ -adic Hensel lifting for algebraic systems. Work in progress, 2012.
- [BP09a] C. Beltrán et L. M. Pardo. Efficient polynomial system-solving by numerical methods. *J. Fixed Point Theory Appl.*, 6(1):63–85, 2009.

- [BP09b] C. Beltrán et L. M. Pardo. Smale's 17th problem: Average polynomial time to compute affine and projective solutions. *J. Amer. Math. Soc.*, 22(2):363–385, 2009.
- [BP11a] C. Beltrán et L. M. Pardo. Fast linear homotopy to find approximate zeros of polynomial systems. *Found. Comput. Math.*, 11(1):95–129, 2011.
- [BP11b] J. Berthomieu et L. M. Pardo. Spherical Radon transform and the average of the condition number on certain Schubert subvarieties of a Grassmannian. Manuscript to appear in *J. Complexity*, 2011.
- [Coh46] I. S. Cohen. On the structure and ideal theory of complete local rings. *Trans. Amer. Math. Soc.*, 59:54–106, 1946.
- [CP08] V. Cossart et O. Piltant. Resolution of singularities of threefolds in positive characteristic. I. Reduction to local uniformization on Artin-Schreier and purely inseparable coverings. *J. Algebra*, 320(3):1051–1082, 2008.
- [CP09] V. Cossart et O. Piltant. Resolution of singularities of threefolds in positive characteristic. II. *J. Algebra*, 321(7):1836–1976, 2009.
- [Die68] J. Dieudonné. *Calcul infinitésimal*. Hermann, Paris, 1968.
- [DL08] C. Durvy et G. Lecerf. A concise proof of the Kronecker polynomial system solver from scratch. *Expositiones Mathematicae*, 26(2):101–139, 2008.
- [FS56] A. Fröhlich et J. C. Shepherdson. Effective procedures in field theory. *Philos. Trans. Roy. Soc. London. Ser. A.*, 248:407–432, 1956.
- [Gao03] S. Gao. Factoring multivariate polynomials via partial differential equations. *Math. Comp.*, 72:801–822, 2003.
- [GG03] J. von zur Gathen et J. Gerhard. *Modern computer algebra*. Cambridge University Press, Cambridge, Second edition, 2003.
- [Gir72] J. Giraud. *Étude locale des singularités*. U.E.R. Mathématique, Université Paris XI, Orsay, 1972. Cours de 3ème cycle, 1971–1972, Publications Mathématiques d'Orsay, No. 26.
- [Gir75] J. Giraud. Contact maximal en caractéristique positive. *Ann. Sci. École Norm. Sup. (4)*, 8(2):201–234, 1975.
- [GLS01] M. Giusti, G. Lecerf et B. Salvy. A Gröbner free alternative for polynomial system solving. *J. Complexity*, 17(1):154–211, 2001.
- [G+91] T. Granlund et al. GMP, the GNU multiple precision arithmetic library. 1991. Available from <http://www.swox.com/gmp>.
- [Hal01] E. Hallouin. Computing local integral closures. *J. Symbolic Comput.*, 32(3):211–230, 2001.
- [Hir64] H. Hironaka. Resolution of singularities of an algebraic variety over a field of characteristic zero. I, II. *Ann. of Math. (2)* 79 (1964), 109–203; *ibid. (2)*, 79:205–326, 1964.
- [Hir67] H. Hironaka. Characteristic polyhedra of singularities. *J. Math. Kyoto Univ.*, 7:251–293, 1967.
- [Hir70] H. Hironaka. Additive groups associated with points of a projective space. *Ann. of Math. (2)*, 92:327–334, 1970.
- [H+02] J. van der Hoeven et al. Mathemagix. 2002. Available from <http://www.mathemagix.org>.
- [Hoe94] M. van Hoeij. An algorithm for computing an integral basis in an algebraic function field. *J. Symbolic Comput.*, 18(4):353–363, 1994.
- [Hoe97] J. van der Hoeven. Lazy multiplication of formal power series. Dans W. W. Küchlin, éditeur, *ISSAC '97: Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation*, pages 17–20. Maui, Hawaii, 1997.

- [Hoe02] J. van der Hoeven. Relax, but don't be too lazy. *J. Symbolic Comput.*, 34(6):479–542, 2002.
- [KPS01] T. Krick, L. M. Pardo et M. Sombra. Sharp estimates for the arithmetic Nullstellensatz. *Duke Math. J.*, 109(3):521–598, 2001.
- [Lan02] S. Lang. *Algebra*, volume 211 de *Graduate Texts in Mathematics*. Springer-Verlag, New York, third edition, 2002.
- [Lec07] G. Lecerf. Improved dense multivariate polynomial factorization algorithms. *J. Symbolic Comput.*, 42(4):477–494, 2007.
- [Lec08] G. Lecerf. Fast separable factorization and applications. *Appl. Algebr. Engrg. Comm. Comput.*, 19(2):135–160, 2008.
- [Lec10] G. Lecerf. New recombination algorithms for bivariate polynomial factorization based on Hensel lifting. *Appl. Algebr. Engrg. Comm. Comput.*, 21(2):151–176, 2010.
- [McK01] D. McKinnon. An arithmetic analogue of Bézout's theorem. *Compositio Math.*, 126(2):147–155, 2001.
- [Ost21] A. M. Ostrowski. Über die Bedeutung der Theorie der konvexen Polyeder für die formale Algebra. *Jahresber. Deutsch. Math.-Verein.*, 30(2):98–99, 1921. Talk given at *Der Deutsche Mathematikertag vom 18–24 September 1921 in Jena*.
- [Ost75] A. M. Ostrowski. On multiplication and factorization of polynomials. I. Lexicographic orderings and extreme aggregates of terms. *Aequationes Math.*, 13(3):201–228, 1975.
- [Ost99] A. M. Ostrowski. On the significance of the theory of convex polyhedra for formal algebra. *SIGSAM Bull.*, 33(1):5, 1999. Translated from [Ost21].
- [Rub02] B. Rubin. Inversion formulas for the spherical Radon transform and the generalized cosine transform. *Adv. in Appl. Math.*, 29(3):471–497, 2002.
- [Sha94] I. R. Shafarevich. *Basic algebraic geometry. 1*. Springer-Verlag, Berlin, Second edition, 1994. Varieties in projective space, Translated from the 1988 Russian edition and with notes by Miles Reid.
- [Shu09] M. Shub. Complexity of Bézout's theorem. VI. Geodesics in the condition (number) metric. *Found. Comput. Math.*, 9(2):171–178, 2009.
- [SS93a] M. Shub et S. Smale. Complexity of Bézout's theorem. I. Geometric aspects. *J. Amer. Math. Soc.*, 6(2):459–501, 1993.
- [SS93b] M. Shub et S. Smale. Complexity of Bézout's theorem. II. Volumes and probabilities. Dans *Computational algebraic geometry (Nice, 1992)*, volume 109 de *Progr. Math.*, pages 267–285. Birkhäuser Boston, Boston, MA, 1993.
- [Tra84] B. M. Trager. *Integration of algebraic functions*. PhD thesis, Massachusetts Institute of Technology, 1984.
- [Wal78] R. J. Walker. *Algebraic curves*. Springer-Verlag, New York, 1978. Reprint of the 1950 edition.
- [Wei10] M. Weimann. A lifting and recombination algorithm for rational factorization of sparse polynomials. *J. Complexity*, 26(6):608–628, 2010.

# Chapitre I

## Computing Hironaka's invariants: Ridge and Directrix

### Abstract

In this chapter we present Hironaka's invariants as developed by Giraud: the ridge and the directrix. We give an effective definition and a functorial one and show their equivalence. The fruit is an effective algorithm that computes the additive generators of the "ridge", and the generators of its invariant algebra. This chapter is based on published article written with P. Hivert and H. Mourtada [BHM10].

### 1 Introduction

The problem of the resolution of singularities has made a tremendous progress thanks to Hironaka's contribution. In this chapter, we want to present some objects that he introduced to resolve singularities, in particular we compute the subtle invariant: the *ridge* (The notion "ridge" is "faîte" in the original (French) literature). Take an ideal  $I \subset R$ , for instance  $R$  a polynomial ring (or a localization thereof) over any field. Take  $x \in \text{Spec}(R/I)$ . The directrix and the ridge live in the tangent cone at  $x$ . The directrix is a vector space, the ridge an additive group. These two objects are given only by the *class of isomorphisms* of  $R/I$ . Even more, these invariants "commute with smooth morphisms" [Gir75]. In particular, for any isomorphism:

$$\phi: R/I \longrightarrow S/J,$$

both  $R/I$  and  $S/J$  have isomorphic tangent cone, directrix and ridge at  $x$  and  $\phi(x)$ .

Giraud shows in [Gir75] that the ridge is the tangent cone of a "maximal contact variety" (see [Kol07]). The ridge as we will see is generated by *additive* polynomials. In characteristic 0, this means that the ridge is a linear space, therefore a "maximal contact variety" is smooth. In characteristic  $p > 0$ , additive polynomials may not be linear, therefore the ridge may not be linear and a "maximal contact variety" may not be smooth. This is the crucial fact why Hironaka's proof is not generalized for free to positive characteristic. This generates a major difficulty, still not overcome in the desingularization problem. Another difficulty is that if you blow up a singular variety  $\mathcal{V}$  along a singular point  $x \in \mathcal{V}$ , the points "near" to  $x$  are on the Proj of the

ridge of the tangent cone. In [Hir70], Hironaka shows that, in characteristic  $p > 0$  there are examples of points “near” to  $x$  which are not on the Proj of the directrix of the tangent cone. In the 70's a large literature about “Hironaka's groups” appeared: people has tried to classify the cases where “near” points are not on the Proj of the directrix of the tangent cone. The ridge and “Hironaka's groups” are closely related, but we do not want to say more about this classification problem which is known to be quite difficult. Nowadays, the ridge seems to be forgotten though it is a very interesting object.

The contribution of this chapter is the computation of a basis of the ideal of the ridge whose elements are additive polynomials. Indeed, in [Gir72, Gir75], Giraud shows how to compute a set of generators of this ideal, but they are not additive polynomials in general: see Example I.22. We also hope that we clarified Giraud's proofs.

## Acknowledgment

V. Cossart gave a talk on this topic in Geocrypt and he initiated us in a working group about desingularization in positive characteristic. He is at the origin of this work, we would like to thank him for his helpful remarks. The authors are very grateful to both the referees for their constructive comments about this paper.

## 2 Notation and prerequisites, naive definitions of Ridge and Directrix

Until the end of this chapter,  $\mathbb{K}$  denotes a field of any characteristic. We give in this section an overview about cones, ridges and directrices.

A linear space of dimension  $n$  is  $\mathbb{A}^n := \text{Spec } R$ , where  $R := \mathbb{K}[X_1, \dots, X_n]$ . A cone  $\mathcal{C}$  embedded in  $\mathbb{A}^n$  is given as  $\text{Spec } \mathbb{K}[X_1, \dots, X_n]/I$  where  $I \subset \mathbb{K}[X_1, \dots, X_n]$  is a *homogeneous* ideal.

**Definition I.1. (Directrix)** *The directrix of  $\mathcal{C}$  is the linear space of equations in  $Y_1, \dots, Y_\tau$ , the smallest set of linear forms such that*

$$I = (I \cap \mathbb{K}[Y_1, \dots, Y_\tau]) \mathbb{K}[X_1, \dots, X_n]. \quad (\text{I.1})$$

In a few words, the smallest list of variables to define  $I$ . Geometrically, there are linear subspaces  $W \subset \mathbb{A}^n$  such that  $\mathcal{C} + W = \mathcal{C}$  (take  $W = 0$  for instance), and if  $W_1$  and  $W_2$  are such, then so is  $W_1 + W_2$ . The directrix corresponds to the *biggest* linear subspace  $W$  of  $\mathbb{A}^n$  such that  $\mathcal{C} + W = \mathcal{C}$ .

**Definition I.2. (Naive definition of the ridge)** *The ridge [Hir67] of  $\mathcal{C}$  is the additive space of equations in  $P_1, \dots, P_e$ , the smallest set of additive polynomials such that*

$$I = (I \cap \mathbb{K}[P_1, \dots, P_e]) \mathbb{K}[X_1, \dots, X_n]. \quad (\text{I.2})$$

This definition looks inconsistent, existence is not clear. Consistence is given in Section 3.2. Obviously, they coincide in characteristic 0, but in characteristic  $p > 0$ , they are in general different. In this chapter, following Giraud [Gir72, Gir75], we show that it is easy to compute the ridge (easier than the directrix). Let us note that the ridge has good properties (commutes to base changes, for example) that the directrix has not. For instance, suppose that  $\mathbb{K}$  has characteristic  $p > 0$  and that  $\lambda \in \mathbb{K}$  is not a  $p$ -power, take  $I = (X^p + \lambda Y^p) \mathbb{K}[X, Y]$ , then the directrix is  $V(X, Y)$ , the ridge is  $V(I)$ , where  $V(\mathcal{I})$  stands for the variety defined by ideal  $\mathcal{I}$ . Change  $\mathbb{K}$  in  $\bar{\mathbb{K}}$  its algebraic closure, then the directrix is  $V(X + \sqrt[p]{\lambda} Y)$ , the ridge is still  $V(I)$ .

### 3 The Ridge: formal definition, main properties.

#### 3.1 Ridge as a functor

Let  $\mathbb{A}_{\mathbb{K}}^n$  be the  $n$ -dimensional affine space over  $\mathbb{K}$ . As above let  $\mathcal{C}$  be the cone defined in  $\mathbb{A}_{\mathbb{K}}^n$  by the homogeneous ideal  $I$ , and let  $G$  be the quotient  $R/I$ . The natural  $\mathbb{K}$ -algebra homomorphism

$$\begin{aligned} \Delta: \mathbb{K}[X_1, \dots, X_n] &\longrightarrow \mathbb{K}[X_1, \dots, X_n] \otimes \mathbb{K}[Y_1, \dots, Y_n] \\ X_i &\longmapsto X_i + Y_i \end{aligned}$$

gives  $\mathbb{A}_{\mathbb{K}}^n$  the natural structure of a group scheme. We will call  $+$  the law that it defines. If we see  $\mathbb{A}_{\mathbb{K}}^n$  as its functor of points, then we can define the subfunctor of the category of Schemes over  $\mathbb{K}$  to the category of Sets as follows: for a  $\mathbb{K}$ -Scheme  $S$ ,  $F(S)$  is the subset of of the  $S$ -points  $v$  in  $\mathbb{A}_{\mathbb{K}}^n$  such that  $(v + c) \in \mathcal{C}(S)$  for every  $S$ -point  $c$  of  $\mathcal{C}(S)$ .

Now, we give some consequences of the definition. Let  $S$  be a  $\mathbb{K}$ -Scheme, firstly,  $0$  is a  $S$ -point which lies in  $\mathcal{C}(S)$ , so for all  $v$  in  $F(S)$ ,  $0 + v$  is an element of  $\mathcal{C}(S)$ , that is to say  $F(S) \subset \mathcal{C}(S)$ . Therefore, seen as functors  $F$  is a subset of  $\mathcal{C}$ . Secondly,  $F(S)$  is a group scheme. The  $S$ -point  $0$  lies trivially in  $F(S)$ . Let two  $S$ -points  $v$  and  $w$  in  $F(S)$ , the definition ensures that translations by  $v$  and  $w$  send the cone  $\mathcal{C}(S)$  to itself, so the composition, which is just the translation by  $v + w$  has the same property. This forces  $(v + w)$  to be in  $F(S)$ . Moreover, the inverse of the translation by  $v$ , which is the translation by  $-v$ , preserves  $\mathcal{C}(S)$ , that is to say  $(-v) \in F(S)$ .

**Proposition-Definition I.3.** *The functor  $F$  is representable by a scheme  $F$ . We call this scheme the ridge of  $\mathcal{C}$ .*

The remarks below say that  $F$ , the ridge of  $\mathcal{C}$  is a group scheme, subscheme of  $\mathcal{C}$ , so the ridge of  $F$  (seen as a subscheme of  $\mathcal{C}$ ) is the ridge  $F$ .

**Proof.**

1. Let  $N$  be the maximum degree of a set of generators  $f_1, \dots, f_m$  of  $I$ . Let  $G_\ell$  be the homogeneous component of degree  $\ell$  of  $G$  ( $G$  is a graded algebra because  $I$  is homogeneous). Let  $H := \bigoplus_{\ell \leq N} G_\ell$  the  $\mathbb{K}$ -vector space which is of finite dimension, we can find a  $\mathbb{K}$ -basis of  $H$  formed by monomials  $e_i$ ,  $i \in \Lambda$ . It is easy to compute it,  $f_i = \underline{X}^{A_i} + \sum_{B \in \mathbb{N}^n, B < A_i} \lambda_B \underline{X}^B$ . So  $H$  is spanned by  $\underline{X}^B$ , with  $|B| \leq N$  and  $B \notin \bigcup_{1 \leq i \leq m} A_i + \mathbb{N}^n$ . This family is a basis of  $H$ . Note that  $H$  generates  $G$  as a  $\mathbb{K}$ -algebra.

2. Let  $s$  be the composed morphism

$$s: R \longrightarrow R \otimes_{\mathbb{K}} R \longrightarrow R \otimes_{\mathbb{K}} G,$$

where the first morphism is  $\Delta$  and the second morphism is the canonical one. For every  $d \in \mathbb{N}$ ,  $d \leq N$  and  $f \in I_d$ ,  $s(f)$  is homogeneous of degree  $d$ , therefore  $s(f) \in R \otimes H$  and it can be uniquely written

$$s(f) = \sum_{\ell \in \Lambda} s_{\ell}(f) \otimes e_{\ell}, \quad \text{with } s_{\ell}(f) \in R_{d-\deg(e_{\ell})}.$$

This follows from the fact that  $R \otimes_{\mathbb{K}} H$  is a free  $R$ -module generated by the  $1 \otimes e_{\ell}$ 's. Now,  $f_1, \dots, f_m$  span  $I$ , so we define  $J$  the ideal generated by  $s_{\ell}(f_i)$ ,  $\ell \in \Lambda$ ,  $1 \leq i \leq m$ .

3. The subscheme of  $\mathbb{A}_{\mathbb{K}}^n$  defined by  $J$  represents the functor  $f$ . Indeed, it is sufficient to verify that for a  $\mathbb{K}$ -algebra  $B$ , the functor of points of  $\text{Spec}(R/J)$  applied to  $B$  coincides with  $F(B)$ . The data of a  $B$ -point of  $\mathbb{A}_{\mathbb{K}}^n$  is equivalent to the data of a homomorphism  $v: R \longrightarrow B$ , which gives rise to

$$R \xrightarrow{\Delta} R \otimes_{\mathbb{K}} R \longrightarrow R \otimes_{\mathbb{K}} G \xrightarrow{v \otimes 1} B \otimes G.$$

If we want the translation by  $v$  to map  $\mathcal{C}$  in  $\mathcal{C}$ , i.e. that  $v$  belongs to  $F(B)$ ,  $(v \otimes 1) \circ s$  must annihilate  $I$ . This means that  $I$  should be in the kernel of  $(v \otimes 1) \circ s$  and therefore the image of the translation by  $v$  is included in  $\mathcal{C}$ . This is equivalent to  $(v \otimes 1) \circ s(f) = \sum_{\ell \in \Lambda} v(s_{\ell}(f)) \otimes e_{\ell} = 0$  for every  $f \in I_d$ ,  $d \leq N$ . But since  $B \otimes_{\mathbb{K}} H$  is free of base  $1 \otimes e_{\ell}$ ,  $\ell \in \Lambda$ , this is equivalent to  $v(s_{\ell}(f)) = 0$ , therefore  $v$  factors by  $R/J$  and it is an  $R/J$ -point.  $\square$

Recall that  $F$  is an additive group and there is no reason for the  $s_i(f_j)$ 's to be additive polynomials in the general case. The idea of Giraud is to find a condition on  $f_1, \dots, f_m$  to have this property.

We define by the Taylor formula, derivations of  $f$ , homogeneous polynomial of degree  $s$ ,  $D_A^X f$  with  $A \in \mathbb{N}^n$  by  $f(\underline{X} + \underline{Y}) = \sum_{A \in \mathbb{N}^n, |A| \leq s} D_A^X f(\underline{X}) \underline{Y}^A$ . This derivations  $D_A^X$  are known as "Hasse-Schmidt" derivations.

**Notations I.4.** *From now on, we will only use the graded lexical order (grlex). Hence, "Gröbner basis" will always mean "Gröbner basis with respect to the grlex order". The ideal of the ridge will be denoted by  $J$ .*

*For any  $P = \sum_{A \in \mathbb{N}^n} \lambda_A \underline{X}^A \in \mathbb{K}[\underline{X}]$ ,  $P \neq 0$ ,  $\exp(P)$  is the greatest  $A$  such that  $\lambda_A \neq 0$ .*

*For any homogeneous ideal  $I \neq \{0\}$  in  $\mathbb{K}[\underline{X}]$ , the set  $\{\exp(P): P \in I \setminus \{0\}\}$  is denoted  $\exp(I)$  and is called the exponent of the ideal  $I$ .*

**Corollary I.5. (Giraud)** *If  $f_1, \dots, f_m$ , the homogeneous generators of  $I$ , satisfy  $D_A^X f_i = 0$  with  $A \in \exp(I)$  and  $|A| < \deg(f_i)$ , then  $J$  is spanned by the  $D_A^X f_i$ 's with  $A \in \mathbb{N}^n$ ,  $|A| < \deg(f_i)$ .*

**Proof.** We keep the same notations as in Proposition-Definition I.3 and we identify  $R \otimes_{\mathbb{K}} R$  with  $\mathbb{K}[\underline{X}, \underline{Y}]$ . Let  $\bar{Y}_i$  be the class of  $Y_i$  in  $R \otimes_{\mathbb{K}} R/I$ . Since the  $Y_i^{A_i}$ 's,  $A_i \notin \exp(I)$ , represent a  $\mathbb{K}$ -basis of  $R/I$ , the  $\bar{Y}_i^{A_i}$ 's,  $A_i \notin \exp(I)$ , give a basis of the free  $R$ -module  $R \otimes_{\mathbb{K}} R/I$ . So with respect to this basis using the Taylor formula, we have that the  $s_{\ell}(f)$ 's, defined as above, are the  $D_A^X f_i$ 's, when the  $f_i$ 's are as above.  $\square$

**Definition I.6.** A basis of  $I$  which verifies the statement of Corollary I.5 will be called a Giraud basis of the cone.

By the definition of the Hasse-Schmidt derivations above and for  $f \in R$  we have

$$f(\underline{X} + \underline{Y}) - f(\underline{X}) - f(\underline{Y}) = \sum_{0 < |A| < d} (D_A^X f)(\underline{X}) \underline{Y}^A. \quad (\text{I.3})$$

**Remark I.7.** We consider

$$U = \{f \in R \mid f(\underline{X} + \underline{Y}) - f(\underline{Y}) \in J \otimes_{\mathbb{K}} \mathbb{K}[\underline{Y}]\}. \quad (\text{I.4})$$

Clearly this is a subalgebra of  $R$ , and it is the invariant algebra of the ridge in  $\mathbb{A}_{\mathbb{K}}^n$ .

Indeed, since the diagonal morphism from  $R$  to  $R \otimes_{\mathbb{K}} R$  identifies with

$$\begin{aligned} \mathbb{K}[\underline{X}] &\longrightarrow \mathbb{K}[\underline{X} + \underline{Y}] \\ f(\underline{X}) &\longmapsto f(\underline{X} + \underline{Y}), \end{aligned}$$

then  $U$  is the algebra of functions on  $\mathbb{A}_{\mathbb{K}}^n$  such that for every  $\mathbb{K}$ -scheme  $S$  and every  $S$ -point  $(u, v)$  of  $F \times_{\mathbb{K}} \mathbb{A}_{\mathbb{K}}^n$ , we have  $f(u + v) = f(u)$ . Let's call  $\Pi$  the following morphism

$$\begin{aligned} \Pi: R &\longrightarrow R \otimes_{\mathbb{K}} R \longrightarrow R/J \otimes_{\mathbb{K}} R \\ f(\underline{X}) &\longmapsto f(\underline{X}) \longmapsto f(\underline{X}). \end{aligned}$$

Elements of  $U$  are those whose images by  $\Delta$  and by  $\Pi$  are the same, hence it is the kernel of  $\Delta - \Pi$ . This means it is the kernel of the double morphism  $(\Delta, \Pi)$ . Since  $R$  has a graded structure, it inherits also a graded structure and from formulae (I.3) and (I.4), for  $d \in \mathbb{N}$  we have

$$U_d = \{f \in J_{d'} \mid D_A^X f \in J, d \leq d', d = d' - |A|\}. \quad (\text{I.5})$$

By Taylor formula, for all  $f \in U_d$  and for all multi-indices  $A$ ,  $|A| \leq d$ ,  $D_A^X f \in U_{d-|A|}$ .

**Lemma I.8.** With the same notations as above, let  $H$  be a  $\mathbb{K}$ -graded subalgebra of  $\mathbb{K}[\underline{X}]$ , then the following assertions are equivalent:

1. For all  $f \in H_d$ , for all multi-index  $A$ ,  $|A| \leq d$ ,  $D_A^X f \in H_{d-|A|}$ ;
2. There exist additive homogeneous polynomials  $\theta_1, \dots, \theta_s, \dots$  such that

$$H = \mathbb{K}[\theta_1, \dots, \theta_s, \dots].$$

Furthermore, in positive characteristic  $p$ , if the conditions above are fulfilled, up to a re-indexation of the variables, one can take

$$\theta_i = X_i^{p^{\alpha_i}} + t_i(X_{i+1}, \dots, X_n), \quad 1 \leq i \leq s < \infty, \quad (\text{I.6})$$

$\alpha_i \leq \alpha_{i+1}$ ,  $1 \leq i \leq s-1$  and  $t_i$ , additive polynomials, in  $\mathbb{K}[X_{i+1}, \dots, X_n]$ .

**Proof.** For  $1 \Rightarrow 2$ , we follow Giraud's idea [Gir72, p. I-29, 30]. Let  $K$  be the subalgebra of  $H$  generated by all additive homogeneous polynomials. Let  $N \in \mathbb{N}$  such that  $H_d = K_d$  for all  $d < N$ . Let  $f = \sum_{A, |A|=N} f_A \underline{X}^A \in H_N$ ,  $f_A \in \mathbb{K}$ , we will prove that  $f \in K_N$ . Let  $K_{N-}$  be the algebra generated by  $K_0, \dots, K_{N-1}$ , we will prove that  $f$  is the sum of elements in  $K_{N-}$  and of an additive polynomial.

Let  $A$  be the greatest multi-index for lex such that  $\underline{X}^A$  is not additive. If  $A$  is in  $\exp(K_{N-})$ , let  $g$  be a polynomial in  $K_{N-}$  such that its greatest monomial for lex is  $\underline{X}^A$ . Then the greatest monomial of  $(f - f_A g)$  is a  $\underline{X}^B$  with  $B < A$ . By induction, we can find a suitable  $f$  such that its greatest non additive monomial for lex is not in  $\exp(K_{N-})$ . We may assume that  $A$  is not the exponent (see Notations I.4) of an element of  $K_{N-}$ ,  $A := (a_1, \dots, a_n)$ ,  $a_i = p^{\beta_i} q_i$ ,  $q_i$  relatively prime to  $p$ . There exist multi-indices  $C$  and  $D$  such that  $A = C + D$ ,  $D_C^X \underline{X}^A \neq 0$  and  $D_D^X \underline{X}^A \neq 0$ . Indeed, either there are two indices  $1 \leq i_0 < i_1 \leq n$  such that  $a_{i_0} a_{i_1} \neq 0$ , take  $C = A - (0, \dots, 0, i_0, 0, \dots, 0)$ ,  $D = (0, \dots, 0, i_0, 0, \dots, 0)$ , either there is only one index  $1 \leq i_0 \leq n$  with  $a_{i_0} \neq 0$  and  $q_{i_0} > 1$ , take  $C = (0, \dots, 0, p^{\beta_{i_0}}, 0, \dots, 0)$  and  $D = (0, \dots, 0, p^{\beta_{i_0}}(q_{i_0} - 1), 0, \dots, 0)$ . We have  $D_C^X \underline{X}^A D_D^X \underline{X}^A = a \underline{X}^A$ ,  $a \in \mathbb{K}^*$ . By hypothesis,  $D_C^X f \in H_{N-|C|} = K_{N-|C|}$  and  $D_D^X f \in H_{N-|D|} = K_{N-|D|}$ ,  $A$  is the exponent of  $D_C^X \underline{X}^A D_D^X \underline{X}^A \in K_{N-}$ . This is a contradiction and  $A$  does not exist, hence  $f$  is additive.

Let us prove the converse. We denote  $g = \sum \lambda_B \theta^B \in \mathbb{K}[\theta_1, \dots, \theta_s, \dots]$ . We have

$$\begin{aligned} g(\underline{X} + \underline{X}') &= \sum \lambda_B \theta(\underline{X} + \underline{X}')^B \\ &= \sum \lambda_B (\theta(\underline{X}) + \theta(\underline{X}'))^B \\ &= \sum \lambda_B \binom{B}{B'} \theta(\underline{X})^{B-B'} \theta(\underline{X}')^{B'} \\ &= \sum_C P_C(\theta(\underline{X})) \underline{X}'^C, \end{aligned}$$

with  $P_C \in \mathbb{K}[\theta_1, \dots, \theta_s, \dots]$ .

The next lemma applied to  $I = H_{>0}$  ends the proof.  $\square$

**Lemma I.9.** *Let  $\text{char } \mathbb{K} = p > 0$ , with notations as above, let  $K$  be a  $\mathbb{K}$ -graded subalgebra of  $\mathbb{K}[\underline{X}]$ , and  $I$  be an ideal generated by a set of additive homogeneous polynomials  $\phi_1, \dots, \phi_m, \dots$ , then, up to a re-indexation of the variables, we can take*

$$\theta_i = X_i^{p^{\alpha_i}} + t_i(X_{i+1}, \dots, X_n), \quad 1 \leq i \leq s \leq n < \infty,$$

$\alpha_i \leq \alpha_{i+1}$ ,  $1 \leq i \leq s-1$  and  $t_i$ , additive polynomials, in  $\mathbb{K}[X_{i+1}, \dots, X_n]$ .

**Proof.** We may assume  $\deg(\phi_i) \leq \deg(\phi_{i+1})$ ,  $1 \leq i \leq m-1$ . By making linear combinations among the  $\phi_i$  of smallest degree, up to a re-indexation of the variables, we may assume that

$$\phi_i = X_i^{p^{\alpha_i}} + t_i(X_{i+1}, \dots, X_n),$$

with  $\mu_i \neq 0$ ,  $\phi_i$  of smallest degree.

We may assume formula (I.6) for every  $\phi_i$ . Indeed, let  $i_0$  be the smallest index such that we have not this formula for  $\phi_{i_0}$ , then

$$\phi_{i_0} = \sum_{1 \leq j \leq m} \mu_{i_0, j} X_j^{p^{\alpha_{i_0}}},$$

where  $\mu_{i_0, j} \in \mathbb{K}$ . Assume for instance that  $\mu_{i_0, 1} \neq 0$ , then we change  $\phi_{i_0}$  in

$$\phi_{i_0, 1} := \phi_{i_0} - \frac{\mu_{i_0, 1}}{\mu_1} \phi_1^{p^{\alpha_{i_0} - \alpha_1}} \in \mathbb{K}[X_2, \dots, X_n],$$

by an easy induction, we change  $\phi_{i_0}$  in  $\phi_{i_0, i_0-1} \in \mathbb{K}[X_{i_0}, \dots, X_n]$ , the reader ends the claim.  $\square$

**Corollary I.10.** *Let  $U$  be  $\mathbb{K}[\theta_1, \dots, \theta_s]$ , then it is a polynomial algebra of variables  $\theta_1, \dots, \theta_s$ .*

**Proof.** Left to the reader.  $\square$

**Corollary I.11.** *With notations as above,  $R$  is a free module over  $U$  of basis*

$$\underline{X}^A, A = (a_1, \dots, a_n), a_i < p^{\alpha_i}, 1 \leq i \leq s.$$

Indeed, if  $\exp(\underline{X}^A \underline{\theta}^B) = \exp(\underline{X}^{A'} \underline{\theta}^{B'})$  with  $A = (a_1, \dots, a_n)$ ,  $a_i < p^{\alpha_i}$ ,  $1 \leq i \leq s$ ,  $A' = (a'_1, \dots, a'_n)$ ,  $a'_i < p^{\alpha_i}$ ,  $1 \leq i \leq s$ ,  $B, B' \in \mathbb{N}^n$ , by formula (I.6),  $(A, B) = (A', B')$ . So the set of  $\underline{X}^A$  is  $U$ -free.

Furthermore,

$$\{\exp(\underline{X}^A \underline{\theta}^B) \mid A = (a_1, \dots, a_n), a_i < p^{\alpha_i}, 1 \leq i \leq s, B \in \mathbb{N}^n\} = \mathbb{N}^n,$$

the set of  $\underline{X}^A$  generates  $S$  over  $U$ .

**Proposition I.12.** *Let  $(f_1, \dots, f_m)$  be a Giraud basis of  $I$ . The  $D_A^X f_i$ 's for  $|A| < \deg f_i$ ,  $i = 1, \dots, m$  generate  $U$ .*

**Proof.** Let  $V$  be the subalgebra of  $R$  generated by the  $D_A^X f_i$ 's for  $|A| < \deg f_i$ ,  $i = 1, \dots, m$ . Since  $U$  is as in formula (I.5),  $V \subset U$ . The polynomials  $D_A^X f_i$  are homogeneous, so  $V$  is a graded subalgebra of  $U$ . Denote by  $U_+$  and  $V_+$  the ideals  $\bigoplus_{d>0} U_d$  and  $\bigoplus_{d>0} V_d$ . From Corollary I.5, we have that  $V_+ R = J$  therefore  $U_+ R = V_+ R = J$ . On the other hand since  $R$  is faithfully flat over  $U$  (see Corollary I.11), we have that  $V_+ U = U_+$ . And we deduce by induction on the degree that  $V = U$ .  $\square$

## 3.2 Naive and formal definitions coincide

**Proposition I.13.** *Let  $J \subset \mathbb{K}[X_1, \dots, X_n]$  be a homogeneous ideal generated by additive polynomials, then there exists  $\mathcal{G} := \{\phi_1, \dots, \phi_s\}$ , a reduced homogeneous Gröbner basis of  $J$ , such that, up to a re-indexation of the variables,*

$$\phi_i = \mu_i X_i^{p^{\alpha_i}} + t_i(X_{i+1}, \dots, X_n), \quad (\text{I.7})$$

with  $\mu_i \neq 0$ ,  $1 \leq i \leq s$ ,  $\alpha_i \leq \alpha_{i+1}$ ,  $1 \leq i \leq s - 1$  and  $t_i$ , additive polynomials, in  $\mathbb{K}[X_{i+1}, \dots, X_n]$ .

Furthermore, up to a re-indexation of the variables, formula (I.7) is true for all reduced homogeneous Gröbner bases of  $J$ .

**Proof.** The first assertion is a direct consequence of Lemma I.9: it is clear that a set of generators verifying formula (I.6) is a reduced homogeneous Gröbner basis of  $J$ .  $\square$

**Corollary I.14.** *Let  $I$  be a homogeneous ideal of  $\mathbb{K}[X_1, \dots, X_n]$ , let  $\mathcal{G} := \{\gamma_1, \dots, \gamma_s\}$  be any reduced homogeneous Gröbner basis of  $J$  the ideal of the ridge of  $V(I)$ , then*

$$I = (I \cap \mathbb{K}[\gamma_1, \dots, \gamma_s]) \mathbb{K}[X_1, \dots, X_n],$$

$U = \mathbb{K}[\gamma_1, \dots, \gamma_s]$  and if  $K$  is a  $\mathbb{K}$ -algebra generated by additive polynomials such that

$$I = (I \cap K) \mathbb{K}[X_1, \dots, X_n], \quad (\text{I.8})$$

then  $U \subset K$ .

**Proof.** Let  $(f_1, \dots, f_m)$  be a Giraud basis of  $I$ , by Corollary I.5, the  $f_i$ 's generate  $U$ , so Proposition I.12 forces that there exists a reduced Gröbner basis  $(\theta_1, \dots, \theta_s)$  of  $J$  whose the form is

$$\theta_i = X_i^{p_i} + t_i(X_{i+1}, \dots, X_n).$$

It follows that  $(\theta_1, \dots, \theta_s)$  is a basis of  $U$  as a  $\mathbb{K}$ -algebra. Now, the particular case  $A=0$  gives that the  $f_i$ 's are elements of  $U$ , so  $I = (I \cap K) \mathbb{K}[X_1, \dots, X_n]$ .

Furthermore, as the ridge of  $J$  is  $J$ , if  $\mathcal{G} := \{\mu_1, \dots, \mu_s\}$  is any reduced homogeneous Gröbner basis of  $J$ , Corollary I.5 and Proposition I.12 applied to  $\mathcal{G}$  give that  $U = \mathbb{K}[\mu_1, \dots, \mu_s]$ .

Let  $K$  be a  $\mathbb{K}$ -algebra generated by additive polynomials such that

$$I = (I \cap K) \mathbb{K}[X_1, \dots, X_n].$$

We can find a basis  $(g_1, \dots, g_s)$  of  $I$ , with  $g_i \in K$ , and then by Corollary I.5, the  $D_A^X g_i$ 's, with  $|A| < \deg f_i$ , generate  $U$ . But Lemma I.8 ensures that this derivations are in  $K$ . Finally,  $U \subset K$ .  $\square$

**Proposition I.15.** *There is a one-to-one correspondence between algebras generated by homogeneous additive polynomials included in  $\mathbb{K}[\underline{X}]$  and ideals generated by homogeneous additive polynomials of  $\mathbb{K}[\underline{X}]$ .*

$$\begin{array}{ccc} \left\{ \begin{array}{l} \text{algebras generated by} \\ \text{homogeneous additive} \\ \text{polynomials} \end{array} \right\} & \longleftrightarrow & \left\{ \begin{array}{l} \text{ideals generated by} \\ \text{homogeneous additive} \\ \text{polynomials} \end{array} \right\} \\ A & \longrightarrow & A_+ \mathbb{K}[X] \\ \mathbb{K}[X]^{\vee(J)} & \longleftarrow & J. \end{array}$$

*This correspondence preserves the inclusion.*

**Example I.16.** Let us explain the correspondence with an example in an algebraic closed field of characteristic 3. Denote by  $U$  the algebra generated by  $X^3$  and  $Y^3 + Z^3$ . It is clear that the ideal  $J$ , image of  $U$  by the first arrow, is spanned by these polynomials.

For the reverse, it is enough to find homogeneous additive polynomials in the algebra (as in the proof of Lemma I.8). Let such a polynomial  $P = \alpha X^{3^a} + \beta Y^{3^a} + \gamma Z^{3^a}$  be in this algebra. We have

$$P(\underline{X} + \underline{X}') - P(\underline{X}') = \alpha X^{3^a} + \beta Y^{3^a} + \gamma Z^{3^a}.$$

So the condition  $P(\underline{X} + \underline{X}') - P(\underline{X}') \in J \otimes \mathbb{K}[\underline{X}']$  implies  $\beta = \gamma$  that is to say  $P = \alpha X^{3^a} + \beta(Y^3 + Z^3)^a$ . This algebra is also equal to  $U$ .

**Proof.** The first arrow is well-defined. The construction of the second arrow is a consequence of Lemma I.9 and Corollary I.10. The bijection is easy to verify.  $\square$

**Corollary I.17.** *Let  $I_1$  and  $I_2$  be homogeneous ideals of  $\mathbb{K}[X_1, \dots, X_n]$ , the following assertions are equivalent:*

1. *The ridge of  $I_2$  contains (as a subscheme) the ridge  $J_1$  of  $I_1$ ;*

2.  $I_2 = (I_1 \cap \mathbb{K}[\theta_1, \dots, \theta_s]) \mathbb{K}[X_1, \dots, X_n]$ , where  $\mathcal{G} := (\theta_1, \dots, \theta_s)$  is any reduced homogeneous Gröbner basis of  $J_1$ .

**Proof.** Left to the reader. □

Now the reader should be convinced that the naive definition I.2 and the formal definition I.3 of the *ridge* coincide.

## 4 An algorithm to compute the ridge and the directrix

### 4.1 An algorithm to compute a “Giraud basis” of the cone

We want to point out that a “Giraud basis” is far from a “reduced Gröbner basis”. Let us give an example to explain it.

**Example I.18.**  $I = (f_1, f_2) \subset \mathbb{K}[X, Y]$  where  $f_1 = XY$ ,  $f_2 = X^3 + Y^3$ . Then  $(f_1, f_2)$  is a “Giraud basis” and not a “reduced Gröbner basis”,  $(f_1, f_2, f_3 = Y^4)$  is a “reduced Gröbner basis”.

**Remark I.19.** A reduced Gröbner basis of the cone truncated to the degree of the greatest given generator is a “Giraud basis”.

We use this easy remark. Our algorithm for computing a “Giraud basis” is almost a Gröbner basis algorithm except we trash out any computed S-polynomial whose degree is greater than the greatest given generator. Actually, since we can know the degree of a S-polynomial before calculating it (recall all our polynomials are homogeneous), if the degree doesn’t match our condition, we skip the computing part. Although they have not been implemented, any known improvement for computing a Gröbner basis, such as in [Laz83, BFS04], can be used in this algorithm.

#### Algorithm I.1

##### Giraud basis algorithm

**Input.** Homogeneous polynomials  $f_1, \dots, f_m$ , such that  $\deg f_1 \leq \dots \leq \deg f_m$ , generating  $I$ .

**Output.** Homogeneous polynomials  $g_1, \dots, g_r$ , such that  $\deg g_i \leq \deg f_m$ , generating  $I$  and verifying Giraud’s Corollary hypotheses.

1. **For**  $i$  **from** 1 **to**  $m$   
 $f_i := f_i / \text{lc}(f_i)$ .
2. Compute a Gröbner basis of  $I$  by trashing the polynomials with higher degrees than  $\deg f_m$ .
3. Minimalize and reduce this basis.
4. **Return** the truncated reduced Gröbner basis.

It should be noted that this kind of algorithm has already been implemented in computer algebra softwares such as SINGULAR [DGPS11].

**Example I.20.** Let  $I = (f_1, f_2) \subset \mathbb{K}[X, Y]$ , where  $f_1 = X$ ,  $f_2 = X^p + Y^p$  and  $p = \text{char } \mathbb{K}$ . As  $f_2$  is additive,  $D_A^{(X,Y)}(f_2) = 0$ , for all  $A$ ,  $|A| < p$ ,  $A \in \exp(I)$ . Then  $(f_1, f_2)$  is a ‘‘Giraud basis’’ and not a truncated ‘‘reduced Gröbner basis’’ as in Example I.18. Let us note that the monomial  $X^p$  which occurs in the expansion of  $f_2$  is in  $\exp(f_1)$ , so our algorithm will make an unnecessary computation and give  $(f_1, Y^p)$  in output.

## 4.2 From the ‘‘Giraud basis’’ to the ridge

Following Giraud’s Corollary I.5, once we computed a Giraud basis  $(f_1, \dots, f_m)$  of the ideal of the tangent cone, we compute the set  $\mathcal{E} := \{D_A^X f_i, 1 \leq i \leq m, |A| < n(i)\}$  of generators of the ideal of the ridge. There are two very different cases:

1.  $\text{char } \mathbb{K} = 0$ ;
2.  $\text{char } \mathbb{K} = p > 0$ .

In case 1, where  $\text{char } \mathbb{K} = 0$ , to compute the ridge (which is also the directrix by Section 2), we propose the following algorithm. Let us note that, in this case, where  $\text{char } \mathbb{K} = 0$ , up to a multiplication by invertibles, the  $D_A^X$ ’s are the usual differential operators, hence in step 2, our algorithm may be apparently improved when we have a good implementation of the  $D_A^X$ ’s.

### Algorithm I.2 Ridge generators in characteristic 0 algorithm

**Input.** Homogeneous polynomials  $f_1, \dots, f_m$  verifying Giraud’s Corollary hypotheses.

**Output.**  $D_A^X f_i$ ’s of degree 1 for all  $i$ ,  $1 \leq i \leq m$ .

1.  $L := \emptyset$ ;
2. **For**  $i$  **from** 1 **to**  $m$ 
  - a.  $g_i := f_i(\underline{X} + \underline{X}')$ .
  - b. **For each** monomial  $\underline{X}'^A$  **in**  $g_i$ 
    - i.  $h := \text{coeff}(g_i, \underline{X}'^A)$ .
    - ii. **If**  $\deg h = 1$  **then**  $L \leftarrow L \cup \{h\}$ .
3. **Return**  $L$ .

The case 2 is the most interesting and the most difficult. By Giraud’s Corollary I.5, up to a change of indices on the variables, there is a basis

$$\mathcal{A}F := \langle \phi_1, \dots, \phi_\tau \rangle,$$

where  $\phi_i = X_i^{p^{q_i}} + \sum_{i+1 \leq j \leq n} \lambda_j X_j^{p^{q_i}}$ , with  $\lambda_j \in \mathbb{K}$ ,  $1 \leq i \leq \tau$ ,  $q_1 \leq q_2 \leq \dots \leq q_\tau$ . There is no hope that  $\mathcal{A}F \subset \mathcal{E}$ , see the example below.

**Lemma I.21.** *With hypotheses and notations as above, let us denote*

$$\mathcal{E}_p := \{ \psi \in \mathcal{E}, \deg(\psi) \text{ is a } p\text{-power} \}.$$

*Then  $\mathcal{E}_p$  generates the ideal of the ridge.*

Let us note that this generalizes the case 1.

**Proof.** We start with an example and a remark.

**Example I.22.**  $I = (f)$ ,  $f = X^p + Y^{p-1}X + Z^p \in \mathbb{K}[X, Y]$ . Then

$$\begin{aligned} \mathcal{E} &= \{ X^p + Y^{p-1}X + Z^p, Y^j X, Y^i, 1 \leq i \leq p-1, 0 \leq j \leq p-1 \}, \\ \mathcal{E}_p &= \{ X, Y, X^p + Y^{p-1}X + Z^p \}, \\ \mathcal{A}F &= \{ X, Y, Z^p \}. \end{aligned}$$

**Remark I.23.** With hypotheses and notations as above, elements of minimal degree of  $J$  are additive polynomials.

Indeed, elements of minimal degree of  $J$  are linear combinations with coefficients in  $\mathbb{K}$  of elements of minimal degree of a set of generators. As  $J$  is generated by additive polynomials (by a general argument or by Proposition I.24 below), these elements are linear combinations of additive polynomials, hence they are additive.

Let us go back to the proof of Lemma I.21. Take any  $\psi_0 \in \mathcal{E}$  of minimal degree such that  $\deg(\psi_0)$  is not a  $p$ -power, let  $d := \deg(\psi_0)$ . Then the ideals of  $R$ , the first generated by  $\psi \in \mathcal{A}F$ , with  $\deg \psi < d$ , and the second generated by  $\phi_1, \dots, \phi_i, n(i) < d$ ,  $n(i)$  maximal, are equal.

Let  $i_1 = \max \{ i, n(i) < d \}$ , thanks to the fact that  $\deg \phi_i > d$  for  $i > i_1$ , one must have  $\psi_0 \in (\phi_1, \dots, \phi_{i_1})$ . Then replace  $\mathcal{A}F$  by  $\mathcal{A}F \setminus \{ \psi_0 \}$  and make an induction on the cardinality of the set of generators.  $\square$

**Proposition I.24.** *Let  $\mathcal{G} := (\theta_1, \dots, \theta_s)$  be a reduced homogeneous Gröbner basis of  $J$  the ideal of the ridge of  $V(I)$ ,  $I$  be a homogeneous ideal of  $\mathbb{K}[X_1, \dots, X_n]$ , with  $\deg(\theta_i) \leq \deg(\theta_{i+1})$ ,  $1 \leq i \leq s-1$ . Then  $\theta_i$  is an additive polynomial for all  $i$ ,  $1 \leq i \leq s$ .*

**Proof.** By contradiction. Let  $\theta_{i_0} \in \mathcal{G}$  with  $i_0$  minimal such that  $\theta_{i_0}$  is not an additive polynomial, let  $d = \deg(\theta_{i_0})$ . Then

$$\theta_{i_0} = \sum_{\substack{B \in \mathbb{N}^n, |B|=d \\ B \neq (0, \dots, p^\alpha, 0, \dots, 0)}} \mu_B \underline{X}^B + \sum_{\substack{C \in \mathbb{N}^n, |C|=d \\ C = (0, \dots, p^\alpha, 0, \dots, 0)}} \mu_C \underline{X}^C,$$

where  $\mu_B \in \mathbb{K}$  and  $\mu_C \in \mathbb{K}$

$$\theta_{i_0} := \tilde{\theta}_{i_0} + \bar{\theta}_{i_0},$$

with  $\tilde{\theta}_{i_0} \neq 0$ ,  $\bar{\theta}_{i_0}$  additive.

Let  $B_0 := \exp(\bar{\theta}_{i_0}) = (b_1, \dots, b_n)$ .

There exists  $B'$  coordinate wise strictly smaller than  $B_0$  such that

$$\psi_0 := D_{B'}^X(\tilde{\theta}_{i_0}) = D_{B'}^X(\theta_{i_0}) \neq 0.$$

Indeed, either there exists  $j$ , such that  $b_j \neq 0$  and  $b_j < |B_0|$ . Then we can take

$$B' = (b_1, \dots, b_{i-1}, 0, b_{i+1}, \dots, b_n)$$

and we have  $D_{B'}^X(\underline{X}^{B_0}) = X_j^{b_j}$  and

$$\psi_0 = \mu_{B_0} X_j^{b_j} + \sum_{\substack{B \neq B_0 \\ (B-B') \in \mathbb{N}^n}} \mu'_B \underline{X}^{B-B'},$$

or  $B_0 = (0, \dots, 0, p^\alpha q, 0, \dots, 0)$  with  $q$  relatively prime to  $p$  and  $q$  is positive. We take

$$B' = (0, \dots, 0, p^\alpha (q-1), 0, \dots, 0),$$

$D_{B'}^X(\underline{X}^{B_0}) = (q-1) X_j$  and

$$\psi_0 = (q-1) \mu_{B_0} X_j + \sum_{\substack{B \neq B_0 \\ (B-B') \in \mathbb{N}^n}} \mu'_B \underline{X}^{B-B'}.$$

As the ridge of the ridge is the ridge,

$$0 \neq \psi_0 \in J.$$

As  $\deg(\psi_0) < \deg(\theta_{i_0})$ ,  $\theta_{i_0}$  is not an element of minimal degree of  $J$ :  $i_0 \geq 2$ . By Corollary I.5,  $\psi_0 \in J$ , so  $\exp(\psi_0) = (B_0 - B') \in \exp(\theta_1, \dots, \theta_{i_0-1})$ , so  $(B_0 - B') \in \exp(\theta_1, \dots, \theta_{i_0-1})$ , which contradicts the reducedness of  $\mathcal{G}$ .  $\square$

### Algorithm I.3

#### Computation of $\theta_i$ 's algorithm

**Input.** Homogeneous polynomials  $f_1, \dots, f_m$  verifying Giraud's Corollary hypotheses.

**Output.**  $D_A^X f_i$ 's of degree a  $p$ -power for all  $i$ ,  $1 \leq i \leq m$ .

1.  $L := \emptyset$ ;
2. **For**  $i$  **from** 1 **to**  $m$ 
  - a.  $g_i := f_i(\underline{X} + \underline{X}')$ .
  - b. **For each** monomial  $\underline{X}'^A$  **in**  $g_i$ 
    - i.  $h := \text{coeff}(g_i, \underline{X}'^A)$ .
    - ii. **If**  $\deg h = p^r$  **then**  $L \leftarrow L \cup \{h\}$ .
3. **Return** a reduced Gröbner basis of  $L$ .

This last algorithm gives us a sequence of  $\theta_i$ 's.

**Remark I.25.** Calling a Gröbner basis algorithm means that all the computation will be done in  $S$  instead of in  $\mathbb{K}[\theta_1, \dots, \theta_s]$ . Using the techniques of Lemma I.8 and Lemma I.9, we can find an algorithm with computations in  $\mathbb{K}[\theta_1, \dots, \theta_s]$ . We do not think we can save a good amount of time nor memory with such an algorithm that would compute the polynomial algebra  $\mathbb{K}[\theta_1, \dots, \theta_s]$  hidden in  $\mathbb{K}[\underline{X}]$ .

**Remark I.26. (Computation of the directrix)** In the case where  $\mathbb{K}$  is perfect, by Definitions I.1 and I.2, the directrix is the reduction of the ridge. Furthermore, the  $\theta_i$ 's,  $1 \leq i \leq s$  are  $p^{\alpha_i}$ -powers, then the ideal of the directrix is

$$\left( p^{\alpha_1} \sqrt{\theta_1}, \dots, p^{\alpha_s} \sqrt{\theta_s} \right).$$

We do not know any direct method to compute it. Indeed Fröhlich and Shepherdson have even shown that testing if an element is a  $p$ th power is not decidable in general, even if the considered ring is effective [FS56, Section 7] (see also the example in [Gat84, Remark 5.10]).

## Bibliography

- [BFS04] M. Bardet, J.-C. Faugère and B. Salvy. On the complexity of Gröbner basis computation of semi-regular overdetermined algebraic equations. In *International Conference on Polynomial System Solving - ICPSS*, pages 71–75. 2004.
- [BHM10] J. Berthomieu, P. Hivert and H. Mourtada. Computing Hironaka's invariants: Ridge and Directrix. In *Arithmetic, Geometry, Cryptography and Coding Theory 2009*, volume 521 of *Contemp. Math.*, pages 9–20. Amer. Math. Soc., Providence, RI, 2010.
- [DGPS11] W. Decker, G.-M. Greuel, G. Pfister and H. Schönemann. SINGULAR 3-1-3 — A computer algebra system for polynomial computations. 2011. Available from <http://www.singular.uni-kl.de>.
- [FS56] A. Fröhlich and J. C. Shepherdson. Effective procedures in field theory. *Philos. Trans. Roy. Soc. London. Ser. A.*, 248:407–432, 1956.
- [Gat84] J. von zur Gathen. Hensel and Newton methods in valuation rings. *Math. Comp.*, 42(166):637–661, 1984.
- [Gir72] J. Giraud. *Étude locale des singularités*. U.E.R. Mathématique, Université Paris XI, Orsay, 1972. Cours de 3ème cycle, 1971–1972, Publications Mathématiques d'Orsay, No. 26.
- [Gir75] J. Giraud. Contact maximal en caractéristique positive. *Ann. Sci. École Norm. Sup. (4)*, 8(2):201–234, 1975.
- [Hir67] H. Hironaka. Characteristic polyhedra of singularities. *J. Math. Kyoto Univ.*, 7:251–293, 1967.
- [Hir70] H. Hironaka. Additive groups associated with points of a projective space. *Ann. of Math. (2)*, 92:327–334, 1970.
- [Kol07] J. Kollár. *Lectures on resolution of singularities*, volume 166 of *Annals of Mathematics Studies*. Princeton University Press, Princeton, NJ, 2007.
- [Laz83] D. Lazard. Gröbner bases, Gaussian elimination and resolution of systems of algebraic equations. In *Computer algebra (London, 1983)*, volume 162 of *Lecture Notes in Comput. Sci.*, pages 146–156. Springer, Berlin, 1983.



# Chapitre II

## Relaxed algorithms for $p$ -adic numbers

### Abstract

Current implementations of  $p$ -adic numbers usually rely on so called *zealous* algorithms, which compute with truncated  $p$ -adic expansions at a precision that can be specified by the user. In combination with Newton-Hensel type lifting techniques, zealous algorithms can be made very efficient from an asymptotic point of view.

In the similar context of formal power series, another so called *lazy* technique is also frequently implemented. In this context, a power series is essentially a stream of coefficients, with an effective promise to obtain the next coefficient at every stage. This technique makes it easier to solve implicit equations and also removes the burden of determining appropriate precisions from the user. Unfortunately, naive lazy algorithms are not competitive from the asymptotic complexity point of view. For this reason, a new *relaxed* approach was proposed by van der Hoeven in the 90's, which combines the advantages of the lazy approach with the asymptotic efficiency of the zealous approach.

In this chapter, we show how to adapt the lazy and relaxed approaches to the context of  $p$ -adic numbers. We report on our implementation in the C++ library ALGEBRAMIX of MATHEMAGIX, and show significant speedups in the resolution of  $p$ -adic functional equations when compared to the classical Newton iteration. This chapter is based on a published article written with J. VAN DER HOEVEN and G. LECERF [BHL11].

### 1 Introduction

Let  $R$  be an *effective commutative ring*, which means that algorithms are available for all the ring operations. Let  $(p)$  be a proper principal ideal of  $R$ . Any element  $a$  of the completion  $R_p$  of  $R$  for the  $p$ -adic valuation can be written, in a non unique way, as a power series  $\sum_{i \geq 0} a_i p^i$  with coefficients in  $R$ . For example, the completion of  $\mathbb{K}[x]$  for the ideal  $(x)$  is the classical ring of power series  $\mathbb{K}[[x]]$ , and the completion of  $\mathbb{Z}$  for any prime integer  $p$  is the ring of  *$p$ -adic integers* written  $\mathbb{Z}_p$ .

In general, elements in  $R_p$  give rise to infinite sequences of coefficients, which cannot be directly stored in a computer. Nevertheless, we can compute with finite but arbitrarily long expansions of  $p$ -adic numbers. In the so called *zealous* approach, the precision  $n$  of the computations must be known in advance, and fast arithmetic can be used for computations in  $R/(p^n)$ . In the *lazy* framework,  $p$ -adic numbers are really promises, which take a precision  $n$  on input, and provide an  $n$ th order expansion on output.

In [Hoe97] appeared the idea that the lazy model actually allows for asymptotically fast algorithms as well. Subsequently [Hoe02], this compromise between the zealous and the naive lazy approaches has been called the *relaxed* model. The main aim of this chapter is the design of *relaxed* algorithms for computing in the completion  $R_p$ . We will show that the known complexity results for power series extend to this setting. For more details on the power series setting, we refer the reader to the introduction of [Hoe02].

## 1.1 Motivation

Completion and deformation techniques come up in many areas of symbolic and analytic computations: polynomial factorization, polynomial or differential system solving, analytic continuation, etc. They make an intensive use of power series and  $p$ -adic integers.

### 1.1.1 Recursive equations

The major motivation for the relaxed approach is the resolution of algebraic or functional equations. Most of the time, such equations can be rewritten in the form

$$Y = \Phi(Y) \tag{II.1}$$

where the indeterminate  $Y$  is a vector in  $R_p^d$  and  $\Phi$  some algebraic or more complicated expression with the special property that

$$(\tilde{y} - y) \in p^n R_p^d \implies (\Phi(\tilde{y}) - \Phi(y)) \in p^{n+1} R_p^d,$$

for all  $y, \tilde{y} \in R_p^d$  and  $n \in \mathbb{N}$ . In that case, the sequence  $0, \Phi(0), \Phi^2(0), \dots$  converges to a solution  $y \in R_p^d$  of (II.1), and we call (II.1) a recursive equation.

Using zealous techniques, the resolution of recursive equations can often be done using a Newton iteration, which doubles the precision at every step [BK78]. Although this leads to good asymptotic time complexities in  $n$ , such Newton iterations require the computation and inversion of the Jacobian of  $\Phi$ , leading to a non trivial dependence of the asymptotic complexity on the size of  $\Phi$  as an expression. For instance, at higher dimensions  $d$ , the inversion of the Jacobian usually involves a factor  $O(d^3)$ , whereas  $\Phi$  may be of size  $O(d)$ . We shall report on such examples in Section 5.

The main *rationale* behind relaxed algorithms is that the resolution of recursive equations just corresponds to a relaxed evaluation of  $\Phi$  at the solution itself. In particular, the asymptotic time complexity to compute a solution has a linear dependence on the size of  $\Phi$ . Of course, the technique does require relaxed implementations for all operations involved in the expression  $\Phi$ . *The essential requirement for a relaxed operation  $\varphi$*  is that  $\varphi(y)_n$  should be available as soon as  $y_0, \dots, y_n$  are known.

### 1.1.2 Elementary operations

A typical implementation of the relaxed approach consists of a library of basic relaxed operations and a function to solve arbitrary recursive equations built up from these operations. The basic operations typically consist of linear operations (such as addition, shifting, derivation, etc.), multiplication and composition. Other elementary operations (such as division, square roots, higher order roots, exponentiation) are easily implemented by solving recursive equations. In several cases, the relaxed approach is not only elegant, but also gives rise to more efficient algorithms for basic operations.

Multiplication is the key operation and Sections 2, 3 and 4 are devoted to it. In situations where relaxed multiplication is as efficient as naive multiplication (*e.g.* in the naive and Karatsuba models), the relaxed strategy is optimal in the sense that solving a recursive equation is as efficient as verifying the validity of the solution. In the worst case, as we will see in Proposition II.6, *relaxed multiplication complexity*  $R(n)$  is  $O(\log n)$  times greater than that of zealous multiplication modulo  $p^n$ . If  $\mathbb{F}_p$  contains many  $2^p$ th roots of unity, then this overhead can be further reduced to  $O(e^{2\sqrt{\log 2 \log \log n}})$  using similar techniques as in [Hoe07b]. In practice, the overhead of relaxed multiplication behaves as a small constant, even though the most efficient algorithms are hard to implement.

In the zealous approach, the division and the square root usually rely on Newton iteration. In small and medium precisions the cost of this iteration turns out to be higher than a direct call to one relaxed multiplication or squaring. This will be illustrated in Section 6: if  $p$  is sufficiently large, then our relaxed division outperforms zealous division.

### 1.1.3 User-friendly interface

An important advantage of the relaxed approach is its user-friendliness. Indeed, the relaxed approach automatically takes care of the precision control during all intermediate computations. A central example is the Hensel lifting algorithm used in the factorization of polynomials in  $\mathbb{Q}[x]$ : one first chooses a suitable prime number  $p$ , then computes the factorization in  $\mathbb{Z}/p\mathbb{Z}[x]$ , lifts this factorization into  $\mathbb{Q}_p[x]$ , and finally one needs to discover how these  $p$ -adic factors recombine into the ones over  $\mathbb{Q}$  (for details see for instance [GG03, Chapter 15]). Theoretically speaking, Mignotte's bound [GG03, Chapter 6] provides us with the maximum size of the coefficients of the irreducible factors, which yields a bound on the precision needed in  $\mathbb{Q}_p$ . Although this bound is sharp in the worst case, it is pessimistic in several particular situations. For instance, if the polynomial is made of small factors, then the factorization can usually be discovered at a small precision. Here the relaxed approach offers a convenient and efficient way to implement adaptive strategies. In fact we have already implemented the polynomial factorization in the relaxed model with success, as we intend to show in detail in a forthcoming paper.

## 1.2 Our contributions

The relaxed computational model was first introduced in [Hoe97] for formal power series, and further improved in [Hoe02, Hoe07b]. In this chapter, we extend the model to more general completions  $R_p$ . Although our algorithms will be represented

for arbitrary rings  $R$ , we will mainly focus on the case  $R = \mathbb{Z}$  when studying their complexities. In Section 2, we first present the relaxed model, and illustrate it on a few easy algorithms: addition, subtraction, and naive multiplications.

In Section 3, we adapt the relaxed product of [Hoe02, Section 4] to  $p$ -adic numbers. We first present a direct generalization, which relies on products of finite  $p$ -expansions. Such products can be implemented in various ways but essentially boil down to multiplying polynomials over  $R$ . We next focus on the case  $R = \mathbb{Z}$  and how to take advantage of fast hardware arithmetic on small integers, or efficient libraries for computations with multiple precision integers, such as GMP [G+91]. In order to benefit from this kind of fast binary arithmetic, we describe a variant that internally performs conversion between  $p$ -adic and 2-adic numbers in an efficient way. We will show that the performance of  $p$ -adic arithmetic is similar to power series arithmetic over  $R/(p)$ .

For large precisions, such conversions between  $p$ -adic and 2-adic expansions involve an important overhead. In Section 4 we present yet another blockwise relaxed multiplication algorithm, based on the fact that  $R_p \simeq R_{p^k}$  for all  $k \geq 1$ . This variant even outperforms power series arithmetic over  $R/(p)$ . For large block sizes  $k$ , the performance actually gets very close to the performance of zealous multiplication.

In Section 5, we recall how to use the relaxed approach for the resolution of recursive equations. For small dimensions  $d$ , it turns out that the relaxed approach is already competitive with more classical algorithm based on Newton iteration. For larger numbers of variables, we observe important speed-ups.

Section 6 is devoted to division. For power series, relaxed division essentially reduces to one relaxed product [Hoe02, Section 3.2.2]. We propose an extension of this result to  $p$ -adic numbers. For medium precisions, our algorithm turns out to be competitive with Newton's method.

In Section 7, we focus on the extraction of  $r$ th roots. We cover the case of power series in small characteristic, and all the situations within  $\mathbb{Z}_p$ . Common transcendental operations such as exponentiation and logarithm are more problematic in the  $p$ -adic setting than in the power series case, since the formal derivation of  $p$ -adic numbers has no nice algebraic properties. In this respect,  $p$ -adic numbers rather behave like floating point numbers. Nevertheless, it is likely that holonomic functions can still be evaluated fast in the relaxed setting, following [Bre76, CC90, Hoe99, Hoe01, Hoe07a]. We also refer to [Kob84, Kat07] for more results about exponentiation and logarithms in  $\mathbb{Q}_p$ .

Algorithms for  $p$ -adic numbers have been implemented in several libraries and computer algebra systems: P-PACK [Wan84], MAPLE, MAGMA, PARI/GP [PAR08], MATHEMATICA [De 04], SAGE [So09], FLINT [HH09], etc. These implementations all use the zealous approach and mainly provide fixed-precision algorithms for  $R = \mathbb{Z}$ . Only SAGE also proposes a lazy interface. However, this interface is not relaxed and therefore inefficient for large precisions.

Most of the algorithms presented in this chapter have been implemented in the C++ open source library ALGEBRAMIX of MATHEMAGIX [H+02] (revision 4791, freely available from <http://www.mathemagix.org>). Although we only report on timings for  $p$ -adic integers, our code provides support for general effective Euclidean domains  $R$ .

## Acknowledgments

We would like to thank the anonymous referees for their helpful comments.

## 2 Data structures and naive implementations

In this section we present the data structures specific to the relaxed approach, and the naive implementations of the ring operations in  $R_p$ .

### 2.1 Finite $p$ -adic expansions

As stated in the introduction, any element  $a$  of the completion  $R_p$  of  $R$  for the  $p$ -adic valuation can be written, in a non unique way, as a power series  $\sum_{i \geq 0} a_i p^i$  with coefficients in  $R$ . Now assume that  $M$  is a subset of  $R$ , such that the restriction of the projection map  $\pi: R \rightarrow R/(p)$  to  $M$  is a bijection between  $M$  and  $R/(p)$ . Then each element  $a$  admits a unique power series expansion  $\sum_{i \geq 0} a_i p^i$  with  $a_i \in M$ . In the case when  $R = \mathbb{Z}$  and  $p \in \mathbb{N} \setminus \{0, 1\}$ , we will always take  $M = \{0, \dots, p-1\}$ .

For our algorithmic purposes, we assume that we are given quotient and remainder functions by  $p$

$$\begin{aligned} \text{quo}(\cdot, p): R &\longrightarrow R \\ \text{rem}(\cdot, p): R &\longrightarrow M, \end{aligned}$$

so that we have

$$a = \text{quo}(a, p) p + \text{rem}(a, p),$$

for all  $a \in R$ .

Polynomials  $\sum_{i=0}^{n-1} a_i p^i \in M[p]$  will also be called *finite  $p$ -adic expansions* of order  $n$ . In fact, finite  $p$ -adic expansions can be represented in two ways. On the one hand, they correspond to unique elements in  $R$ , so we may simply represent them by elements of  $R$ . However, this representation does not give us direct access to the coefficients  $a_i$ . By default, we will therefore represent finite  $p$ -adic expansions by polynomials in  $M[p]$ . Of course, polynomial arithmetic in  $M[p]$  is not completely standard due to the presence of carries.

### 2.2 Classical complexities

In order to analyze the costs of our algorithms, we denote by  $\mathbf{M}(n)$  the cost for multiplying two univariate polynomials of degree  $n$  over an arbitrary ring  $A$  with unity, in terms of the number of arithmetic operations in  $A$ . Similarly, we denote by  $\mathbf{l}(n)$  the time needed to multiply two integers of bit-size at most  $n$  in the classical *binary representation*. It is classical [SS71, CK91, Für07] that  $\mathbf{M}(n) = O(n \log n \log \log n)$  and  $\mathbf{l}(n) = O(n \log n 2^{\log^* n})$ , where  $\log^*$  represents the iterated logarithm of  $n$ . Throughout the chapter, we will assume that  $\mathbf{M}(n)/n$  and  $\mathbf{l}(n)/n$  are increasing. We also assume that  $\mathbf{M}(O(n)) = O(\mathbf{M}(n))$  and  $\mathbf{l}(O(n)) = O(\mathbf{l}(n))$ .

In addition to the above complexities, which are classical, it is natural to introduce  $l_p(n)$  as the time needed to multiply two  $p$ -adic expansions in  $\mathbb{Z}_p$  at order  $n$  with coefficients in the usual binary representation. When using Kronecker substitution for multiplying two finite  $p$ -adic expansions, we have  $l_p(n) = l(n(\log p + \log n))$  [GG03, Corollary 8.27]. We will assume that  $l_p(n)/n$  is increasing and that  $l_p(O(n)) = O(l_p(n))$ .

It is classical that the above operations can all be performed using linear space. Throughout this chapter, we will make this assumption.

### 2.3 The relaxed computational model

For the description of our relaxed algorithms, we will follow [Hoe02] and use a C++-style pseudo-code, which is very similar to the actual C++ implementation in MATHEMAGIX. As in [Hoe02], we will not discuss memory management related issues, which have to be treated with care in real implementations, especially when it comes to recursive expansions (see Section 5 below).

The main class  $\text{Padic}_p$  for  $p$ -adic numbers really consists of a pointer (with reference counting) to the corresponding abstract representation class  $\text{Padic\_rep}_p$ . On the one hand, this representation class contains the computed coefficients  $\varphi: M[p]$  of the number  $a$  up till a given order  $n: \mathbb{N}$  (let us already mention here that  $\varphi$  can eventually be used to store anticipated data, as in the algorithms of Section 3). On the other hand, it contains a “purely virtual method” `next`, which returns the next coefficient  $a_n$ :

```
class  $\text{Padic\_rep}_p$ 
   $\varphi: M[p]$ 
   $n: \mathbb{N}$ 
  virtual next()
```

Following C++-terminology, the purely virtual function `next` is only defined in a concrete representation class which derives from  $\text{Padic\_rep}_p$ . For instance, to construct a  $p$ -adic number from an element in  $M$ , we introduce the type  $\text{Constant\_Padic\_rep}_p$  that inherits from  $\text{Padic\_rep}_p$  (inheritance is represented by the symbol  $\triangleright$ ) in this way:

```
class  $\text{Constant\_Padic\_rep}_p \triangleright \text{Padic\_rep}_p$ 
   $c: M$ 
  constructor ( $\tilde{c}: M$ )
     $c := \tilde{c}$ 
  method next()
    if  $n = 0$  then return  $c$  else return 0
```

In this piece of code  $n$  represents the current precision inherited from  $\text{Padic\_rep}_p$ . The user visible constructor is given by

$$\text{padic}(c: M) \longrightarrow \text{Padic}_p := (\text{Padic}_p) \text{ new Constant\_Padic\_rep}_p(c).$$

This constructor creates a new object of type  $\text{Constant\_Padic\_rep}_p$  to represent  $c$ , after which its address can be casted to the type  $\text{Padic}_p$  of  $p$ -adic numbers. From now on, for the sake of conciseness, we no longer describe such essentially trivial user level functions anymore, but only the concrete representation classes.

It is convenient to define one more public top-level function for the extraction of the coefficient  $a_k$ , given an instance  $a$  of  $\text{Padic}_p$  and a positive integer  $k: \mathbb{N}$ . This function first checks whether  $k$  is smaller than the order  $a.n$  of  $a$ . If so, then  $a_k = (a.\varphi)_k$  is already available. Otherwise, we keep increasing  $a.n$  while calling `next`, until  $a_k$  will eventually be computed. For more details, we refer to [Hoe02, Section 2]. We will now illustrate our computational model on the basic operations of addition and subtraction.

## 2.4 Addition

The representation class for sums of  $p$ -adic numbers, written `Sum_Padic_repp`, is implemented as follows:

```

class Sum_Padic_repp ≽ Padic_repp
  a, b: Padicp
  γ: R
  constructor (ã: Padicp, b̃: Padicp)
    a := ã; b := b̃; γ := 0
  method next()
    t := an + bn + γ
    γ := quo(t, p)
    return rem(t, p)

```

In the case when  $R = \mathbb{Z}$ , we notice by induction over  $n$  that we have  $\gamma \in \{0, 1\}$ , each time that we enter `next`, since  $0 \leq a_n + b_n + \gamma \leq 2p - 1$ . In that case, it is actually more efficient to avoid the calls to `quo` and `rem` and replace the method `next` by

```

method next()
  t := an + bn + γ
  if t < p then
    γ := 0
    return t
  else
    γ := 1
    return t - p

```

**Proposition II.1.** *Given two relaxed  $p$ -adic integers  $a$  and  $b$ , the sum  $a + b$  can be computed up till precision  $n$  using  $O(n \log p)$  bit-operations.*

**Proof.** Each of the additions  $a_k + b_k + \gamma$  and subsequent reductions modulo  $p$  take  $O(\log p)$  bit-operations.  $\square$

## 2.5 Subtraction in $\mathbb{Z}_p$

In general, the subtraction is the same as the addition, but for the special case when  $R = \mathbb{Z}$ , we may use the classical school book method. In our framework, this yields the following implementation:

```

class Sub_Padic_rep_p  $\supseteq$  Padic_p
  a, b: Padic_p
   $\gamma: R$ 
  constructor ( $\tilde{a}: \text{Padic}_p, \tilde{b}: \text{Padic}_p$ )
     $a := \tilde{a}; b := \tilde{b}; \gamma := 0$ 
  method next()
     $t := a_n - b_n - \gamma$ 
    if  $t \geq 0$  then
       $\gamma := 0$ 
      return t
    else
       $\gamma := 1$ 
      return  $t + p$ 

```

**Proposition II.2.** *Given two relaxed  $p$ -adic integers  $a$  and  $b$ , the difference  $a - b$  can be computed up till precision  $n$  using  $O(n \log p)$  bit-operations.*

**Proof.** Each call to the function `next` costs  $O(\log p)$  bit-operations.  $\square$

## 2.6 Naive product

Here we consider the school book algorithm: each coefficient  $(ab)_n$  is obtained from the sum of all products of the form  $a_k b_{n-k}$  plus the carry involved by the products of the preceding terms. Carries are larger than for addition, so we have to take them into account carefully. The naive method is implemented in the following way:

```

class Naive_Padic_rep_p  $\supseteq$  Padic_rep_p
  a, b: Padic_p
   $\gamma$ : a vector with entries in  $R$ , with indices starting at 0.
  constructor ( $\tilde{a}: \text{Padic}_p, \tilde{b}: \text{Padic}_p$ )
     $a := \tilde{a}; b := \tilde{b};$ 
    Initialize  $\gamma$  with the empty vector
  method next()
    Append a zero  $\gamma_n = 0$  at the end of  $\gamma$ 
     $t := 0$ 
    for  $i$  from 0 to  $n$  do
       $s := a_i b_{n-i} + \gamma_i$ 
       $t := t + s$ 
       $\gamma_i := \text{quo}(t, p)$ 
       $t := \text{rem}(t, p)$ 
    return t

```

Let us precise that, in the pseudo-code, the **for** loop means that  $i$  runs over all the integer values from 0 to  $n$  included.

**Proposition II.3.** *Given two relaxed  $p$ -adic integers  $a$  and  $b$ , the product  $ab$  can be computed up till precision  $n$  using  $O(n^2 \lceil \log p \rceil)$  bit-operations.*

**Proof.** We show by induction that, when entering in `next` to compute the coefficient  $(a b)_n$ , the vector  $\gamma$  has size  $n$  and entries in  $M$ . This clearly holds for  $n = 0$ . Assume that the hypothesis is satisfied until a certain value  $n \geq 0$ . When entering `next` the size of  $\gamma$  is increased by 1, so that it will be  $n + 1$  at the end. Then, at step  $i \in \{0, \dots, n\}$  of the loop we have  $s \leq (p-1)^2 + p - 1 = p^2 - p$ . Since  $t \leq p - 1$  it follows that  $s + t \leq p^2 - 1$ , whence  $\gamma_i \leq p - 1$  on exit. Each of the  $O(n^2)$  steps within the loop takes  $O(\log p)$  bit-operations, which concludes the proof.  $\square$

Since hardware divisions are more expensive than multiplications, performing one division at each step of the above loop turns out to be inefficient in practice. Especially when working with hardware integers, it is therefore recommended to accumulate as many terms  $a_i b_{n-i}$  as possible in  $s$  before a division. For instance, if  $p$  fits 30 bits and if we use 64 bits hardware integers then we can do a division every 16 terms.

## 2.7 Lifting the power series product

In this subsection we assume that we are given an implementation of relaxed power series over  $R$ , as described in [Hoe02, Hoe07b]. The representation class is written `Series_repR` and the user level class is denoted by `SeriesR`, in the same way as for  $p$ -adic numbers. Another way to multiply  $p$ -adic numbers relies on the relaxed product in  $R[[p]]$ . This mainly requires a lifting algorithm of  $M[[p]]$  into  $R[[p]]$  and a projection algorithm of  $R[[p]]$  onto  $M[[p]]$ . The lifting procedure is trivial:

```
class Lift_Series_rep_p ⊇ Series_rep_p
  a: Padic_p
  constructor (ã: Padic_p)
    a := ã
  method next()
    return a_n
```

Let `lift` denote the resulting function that converts a  $p$ -adic number  $a: \text{Padic}_p$  into a series in `SeriesR`. The reverse operation, `project`, is implemented as follows:

```
class Project_Padic_rep_p ⊇ Padic_rep_p
  f: Series_p
  γ: R
  constructor (f̃: Series_R)
    f := f̃; γ := 0
  method next()
    t := f_n + γ
    γ := quo(t, p)
    return rem(t, p)
```

Finally the product  $c = a b$  is obtained as `project(lift(a) lift(b))`.

**Proposition II.4.** *Given relaxed  $p$ -adic integers  $a$  and  $b$ , the product  $a b$  can be computed up till precision  $n$  using  $O(\log p + \log n) M(n) \log n$  or  $O(n (\log p + \log n) \log n)$  bit-operations.*

**Proof.** The relaxed product of two power series in size  $n$  can be done with  $O(M(n) \log n)$  operations in  $R$  by [Hoe02, Section 4.3.2, Theorem 6]. In our situation, the size of the integers in the product are in  $O(\log p + \log n)$ . Then, by induction, one can easily verify that the size of the carry  $\gamma$  does not exceed  $O(\log p + \log n)$  during the final projection step. We are done with the first bound.

The second bound is a consequence of the classical Kronecker substitution: we can multiply two polynomials in  $\mathbb{Z}[x]$  of size  $n$  and coefficients of bit-size  $O(\log p)$  with  $O(l(n(\log p + \log n)))$  bit operations [GG03, Corollary 8.27].  $\square$

This strategy applies in full generality and gives a “softly optimal algorithm”. It immediately benefits from any improvements in the power series product. Nevertheless, when  $n$  is not much larger than  $p$ , practical implementations of this method involve a large constant overhead. In the next sections, we will therefore turn our attention to “native” counterparts of the relaxed power series products from [Hoe02, Hoe07b].

## 2.8 Timings

We conclude this section with some timings for our C++ implementation of naive multiplication inside the ALGEBRAMIX package of the MATHEMAGIX system [H+02]. Timings are measured using one core of an INTEL XEON X5450 at 3.0 GHz running LINUX and GMP 5.0.0 [G+91]. As a comparison, we display timings on the same platform, obtained for the MAPLE 14 package PADIC. Precisely, we created two random numbers to precision  $n$ , did their product via the function `evalp`, and then asked for the coefficient of order  $n/2$ . Notice that timings for small precisions are not very relevant for MAPLE because of the overhead due to the interpreter. As another comparison, we report on the performances of PARI/GP version 2.3.5. For all these three cases we observe that asymptotically fast algorithms are not used. In fact PARI/GP carefully implements the zealous strategy on the binary representation modulo  $p^n$ : as expected, such timings are better than ours, but not so much neither. We shall come back in Section 4 on this critical issue of taking better advantage of native binary representations within the lazy model.

$n$	8	16	32	64	128	256	512	1024
Naive_Mul_Padic $_p$	2.9	3.8	8.3	22	68	250	920	3600
MAPLE 14	240	320	520	1200	3500	11000	38000	160000
PARI/GP	0.52	1.0	2.7	8.4	28	99	360	1300

**Table II.1.** Naive products, for  $p = 536870923$ , in microseconds.

## 3 Relaxed product

In this section, we extend the relaxed product of [Hoe02, Section 4.3.1] to more general  $p$ -adic numbers. We also present a special version for  $R = \mathbb{Z}$ , which uses internal base conversions between base 2 and base  $p$ , and takes better advantage of the fast arithmetic in GMP [G+91].

### 3.1 Fast relaxed multiplication algorithm

Let  $a$  and  $b$  denote the two  $p$ -adic numbers that we want to multiply, and let  $c$  be their product. Let us briefly explain the basic idea behind the speed-up of the relaxed algorithm with respect to naive lazy multiplication.

The first coefficient  $c_0$  is simply obtained as the remainder of  $a_0 b_0$  in the division by  $p$ . The corresponding quotient is stored as a carry in a variable  $\gamma$  similar to the one used in `Naive_Mul_Padic_rep_p`. We next obtain  $c_1$  by computing  $a_0 b_1 + a_1 b_0 + \gamma$  and taking the remainder modulo  $p$ ; the quotient is again stored in  $\gamma$ . At the next stage, which basically requires the computation of  $a_0 b_2 + a_1 b_1 + a_2 b_0 + \gamma$ , we do a little bit more than necessary: instead of  $a_1 b_1$ , we rather compute  $(a_1 + a_2 p)(b_1 + b_2 p)$ . For  $c_3$ , it then suffices to compute  $a_0 b_3$  and  $a_3 b_0$  since  $a_1 b_2 + a_2 b_1$  has already been computed as part of  $(a_1 + a_2 p)(b_1 + b_2 p)$ . Similarly, in order to obtain  $c_4$ , we only need to compute  $a_0 b_4$ ,  $a_4 b_0$ ,  $a_3 b_1$  and  $a_1 b_3$ , since  $a_2 b_2$  is already known. In order to anticipate more future computations, instead of computing  $a_3 b_1$ ,  $a_1 b_3$ , we compute  $(a_1 + a_2 p)(b_3 + b_4 p)$  and  $(a_3 + a_4 p)(b_1 + b_2 p)$ .

In Figure II.1 below, the contribution of each  $a_i b_j$  to the product  $c_{i+j}$ , corresponds to a small square with coordinates  $(i, j)$ . Each such square is part of a larger square which corresponds to a product

$$(a_k + \dots + a_{k+2^q-1} p^{2^q-1})(b_l + \dots + b_{l+2^q-1} p^{2^q-1}).$$

The number inside the big square indicates the stage  $k + l$  at which this product is computed. For instance the products

$$(a_3 + a_4 p + a_5 p^2 + a_6 p^3)(b_7 + b_8 p + b_9 p^2 + b_{10} p^3),$$

$$(a_7 + a_8 p + a_9 p^2 + a_{10} p^3)(b_3 + b_4 p + b_5 p^2 + b_6 p^3).$$

correspond to the two  $4 \times 4$  squares marked with 10 inside.

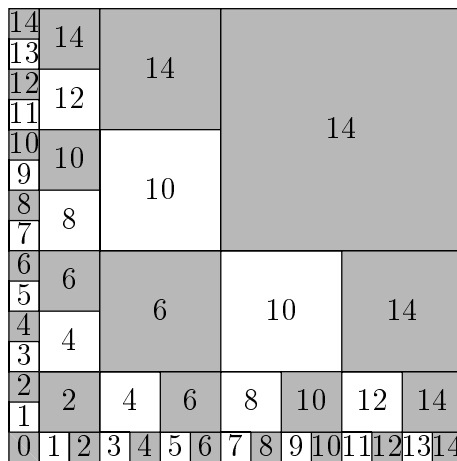


Figure II.1. Relaxed product.

Given a  $p$ -adic number  $a$ , it will be convenient to denote

$$a_{i\dots j} = a_i + a_{i+1} p + \dots + a_{j-1} p^{j-1-i}.$$

For any integer  $n$ , we also define  $l_n$  to be the largest integer such that  $2^{l_n}$  divides  $n+2$  if  $n+2$  is not a power of 2. Otherwise, if  $n+2=2^m$ , we let  $l_n=m-1$ . For instance,  $l_0=0$ ,  $l_1=0$ ,  $l_2=1$ ,  $l_3=0$ ,  $l_4=1$ , etc. In fact, this can be seen in Figure II.1, where the greatest square with number  $n$  inside has size precisely  $2^{l_n} \times 2^{l_n}$ .

We can now describe our fast relaxed product. Recall that  $\varphi$  is the finite  $p$ -expansion inherited from  $\text{Padic\_rep}_p$  that stores the coefficients known to order  $n$ . In the present situation we also use  $\varphi$  for storing the anticipated products.

```

class Relaxed_Mul_Padic_rep  $\triangleright$  Padic_rep_p
  a, b: Padic_p
   $\gamma^a, \gamma^b$ : vectors of vectors over  $R$ , with indices starting from 0

  constructor ( $\tilde{a}$ : Padic_p,  $\tilde{b}$ : Padic_p)
    a :=  $\tilde{a}$ , b :=  $\tilde{b}$ 
    Initialize  $\gamma^a$  and  $\gamma^b$  with the empty vector

  method next()
    On entry,  $\gamma^a$  and  $\gamma^b$  have size  $2n$ ; resize them to  $2(n+1)$ 
    Initialize  $\gamma_{2n}^a$  and  $\gamma_{2n}^b$  with the zero vector of size  $l_{2n}+1$ 
    Initialize  $\gamma_{2n+1}^a$  and  $\gamma_{2n+1}^b$  with the zero vector of
    size  $l_{2n+1}+1$ 

     $t^a := 0, t^b := 0$ 

    for  $q$  from 0 to  $l_n$  do
       $k := (n+2)/2^q$ 
       $t^a += \gamma_{n,q}^a, t^b += \gamma_{n,q}^b$ 
       $t^a += a_{2^q-1\dots 2^{q+1}-1} b_{(k-1)2^q-1\dots k2^q-1}$ 
      if  $k = 2$  then break
       $t^b += a_{(k-1)2^q-1\dots k2^q-1} b_{2^q-1\dots 2^{q+1}-1}$ 

     $s^a := \varphi_{n\dots n+2^{l_n+1}} + t^a$ 
    for  $i$  from 0 to  $2^{l_n+1}-1$  do  $\varphi_{n+i} := s_i^a$ 
    if  $n+2 \neq 2^{l_n+1}$  then  $\gamma_{n+2^{l_n+1}, l_n}^a := s_{2^{l_n+1}}^a$ 

     $s^b := \varphi_{n\dots n+2^{l_n+1}} + t^b$ 
    for  $i$  from 0 to  $2^{l_n+1}-1$  do  $\varphi_{n+i} := s_i^b$ 
    if  $n+2 \neq 2^{l_n+1}$  then  $\gamma_{n+2^{l_n+1}, l_n}^b := s_{2^{l_n+1}}^b$ 

    return  $\varphi_n$ 

```

**Example II.5.** Let us detail how our algorithm works with the first four steps of the multiplication  $c = ab$  with

$$\begin{aligned} a &= 676 = 4 + 5 \times 7 + 6 \times 7^2 + 7^3, \\ b &= -1 = 6 + 6 \times 7 + 6 \times 7^2 + 6 \times 7^3 + O(7^4). \end{aligned}$$

*Computation of  $c_0$ .* Since  $l_0 = l_1 = 0$ , the entries  $\gamma_0^a, \gamma_0^b, \gamma_1^a, \gamma_1^b$  are set to (0). In the **for** loop,  $q$  takes the single value 0, which gives  $k=2$ ,  $t^a = a_0 b_0 = 3 + 3 \times 7$ , and  $t^b = 0$ . Then we deduce  $s^a = 3 + 3 \times 7$ , and we set  $\varphi_0 = 3, \varphi_1 = 3$ . In return we have  $c_0 = 3$ .

*Computation of  $c_1$ .* We have  $l_1 = 0$ ,  $l_2 = 1$  and  $l_3 = 0$ , so that  $\gamma_2^a$  and  $\gamma_2^b$  are initialized with  $(0, 0)$ , while  $\gamma_3^a$  and  $\gamma_3^b$  are initialized with  $(0)$ . In the **for** loop,  $q$  takes again the single value 0, and  $k$  is set to 3. We obtain  $t^a = a_0 b_1 = 3 + 3 \times 7$  and  $t^b = a_1 b_0 = 2 + 4 \times 7$ . It follows that  $s^a = 6 + 3 \times 7$ , and then that  $s^b = 1 + 1 \times 7 + 1 \times 7^2$ . Finally we set  $\varphi_1 = 1$ ,  $\varphi_2 = 1$ ,  $\gamma_{3,0}^b = s_2^b = 1$ , and we return  $c_1 = 1$ .

*Computation of  $c_2$ .* We have  $l_2 = 1$ ,  $l_4 = 1$  and  $l_5 = 0$ , so that  $\gamma_4^a$  and  $\gamma_4^b$  are initialized with  $(0, 0)$  and  $\gamma_5^a$  and  $\gamma_5^b$  with  $(0)$ . During the first step of the **for** loop we have  $q = 0$ ,  $k = 4$ ,  $t^a = a_0 b_2 = 3 + 3 \times 7$  and  $t^b = a_2 b_0 = 1 + 5 \times 7$ . In the second step we have  $q = 1$ ,  $k = 2$ , and we add  $a_{1\dots 3} b_{1\dots 3} = (a_1 + a_2 \times 7) (b_1 + b_2 \times 7) = 2 + 4 \times 7^2 + 6 \times 7^3$  to  $t^a$ , its value becomes  $5 + 3 \times 7 + 4 \times 7^2 + 6 \times 7^3$ . Then we get  $s^a = 6 + 3 \times 7 + 4 \times 7^2 + 6 \times 7^3$ , and then  $s^b = 2 \times 7 + 5 \times 7^2 + 6 \times 7^3$ . Finally we set  $\varphi_2 = 0$ ,  $\varphi_3 = 2$ ,  $\varphi_4 = 5$ ,  $\varphi_5 = 6$ , and return 0 for  $c_2$ .

*Computation of  $c_3$ .* We have  $l_3 = 0$ ,  $l_6 = 2$  and  $l_7 = 0$ , hence  $\gamma_6^a$  and  $\gamma_6^b$  are set to  $(0, 0, 0)$ , and  $\gamma_7^a$  and  $\gamma_7^b$  to  $(0)$ . In the **for** loop,  $q$  takes the single value 0. We have  $k = 5$ ,  $t^a = a_0 b_3 = 3 + 3 \times 7$  and  $t^b = \gamma_{3,0}^b + a_3 b_0 = 7$ . Then we deduce  $s^a = 5 + 7 + 7^2$  which yields  $\gamma_{5,0}^a = 1$ , and then  $s^b = 5 + 2 \times 7$ . In return we thus obtain  $c_3 = 5$ .

**Proposition II.6.** *Given relaxed  $p$ -adic integers  $a$  and  $b$ , the product  $ab$  can be computed up till precision  $n$  using  $O(l_p(n) \log n)$  bit-operations. For this computation, the total amount of space needed to store the carries  $\gamma^a$  and  $\gamma^b$  does not exceed  $O(n)$ .*

**Proof.** The proof is essentially the same as for [Hoe02, Section 4.3.2, Theorem 6], but the carries require additional attention. We shall prove by induction that all the entries of  $\gamma^a$  and  $\gamma^b$  are always in  $\{0, 1\}$  when entering next for the computation of  $\varphi_n$ . This holds for  $n = 0$ . Assume now that it holds for a certain  $n \geq 0$ . After the first step of the loop, namely for  $q = 0$ , we have  $t^a \leq (p-1)^2 + 1 \leq p^2 - 1$ . After the second step, when  $q = 1$ , we have  $t^a \leq (p^2 - 1)^2 + p^2 - 1 + 1 \leq p^4 - 1$ . By induction, it follows that  $t^a \leq p^{2^{q+1}} - 1$ , at the end of the  $q$ th step.

At the end of the **for** loop, we thus get  $t^a \leq p^{2^{l_n+1}} - 1$ . This implies  $s^a \leq 2p^{2^{l_n+1}} - 2$ , whence  $\gamma_{n+2^{l_n+1}, l_n}^a \in \{0, 1\}$ . The same holds for superscripts  $b$  instead of  $a$ . Notice that if  $n + 2 \neq 2^{l_n+1}$  then  $2^{l_n+1} \leq n + 1$ , hence  $n + 2^{l_n+1} \leq 2n + 1$ . This implies that  $\gamma_{n+2^{l_n+1}, l_n}^a$  and  $\gamma_{n+2^{l_n+1}, l_n}^b$  are well defined.

If  $n + 2 = 2^{l_n+1}$ , then  $s_{2^{l_n+1}}^a = s_{2^{l_n+1}}^b = 0$ , since  $s^a$  and  $s^b$  are bounded by  $(p^{n+1} - 1)^2 / p^n < p^{n+2}$ . On the other hand, the map  $n \mapsto (n + 2^{l_n+1}, l_n)$  is injective, so that each entry of  $\gamma^a$  and  $\gamma^b$  can be set to 1 at most once. It thus follows that all the carries are carefully stored in the vectors  $\gamma^a$  and  $\gamma^b$ .

If  $n + 2 = 2^{l_n} u$ , with  $u \geq 2$ , then  $n + 2^{l_n+1} + 2 = 2^{l_n} (u + 2)$ , with  $u + 2 \geq 2$ . This implies that, when we arrive at order  $n' = n + 2^{l_n+1}$ , then the value  $l_{n'}$  is at least  $l_n$ . Therefore all the carries are effectively taken into account. This proves the correctness of the algorithm.

The cost of the algorithm at order  $n$  is

$$O\left(\sum_{i=0}^n \sum_{q=0}^{l_i} l_p(2^i)\right) = O\left(\sum_{2^q \leq n} \frac{n}{2^q} l_p(2^q)\right) = O\left(l_p(n) \sum_{2^q \leq n} 1\right) = O(l_p(n) \log n),$$

using our assumption that  $l_p(n)/n$  is increasing. Finally,

$$O\left(\sum_{i=0}^n (1 + l_i)\right) = O\left(\sum_{i=0}^n \sum_{q=0}^{l_i} 1\right) = O\left(\sum_{2^q \leq n} \frac{n}{2^q}\right) = O(n)$$

provides enough space for storing all the carries.  $\square$

In practice, instead of increasing the sizes of carry vectors by one, we double these sizes, so that the cost of the related memory allocations and copies becomes negligible. The same remark holds for the coefficients stored in  $\varphi$ .

When multiplying finite  $p$ -adic expansions using Kronecker substitution, we obtain a cost similar to the one of Proposition II.4. Implementing a good product for finite  $p$ -adic expansions requires some effort, since we cannot directly use binary arithmetic available in the hardware. In the next subsection, we show that minor modifications of our relaxed product allow us to convert back and forth between the binary representation in an efficient manner. Finally, notice that in the case when  $R = \mathbb{K}[x]$  and  $(p) = (x)$ , the carries  $\gamma^a$  and  $\gamma^b$  are useless.

### 3.2 Variant with conversion to binary representation

In this subsection, we assume that  $R = \mathbb{Z}$  and we adapt the above relaxed product in order to benefit from fast binary arithmetic available in the processor or GMP. In fact we shall convert from base  $p$  to base 2 in order to perform most of the internal computations efficiently, but backward conversions are needed for the output. These conversions can be naturally integrated in an efficient manner as described in the following algorithm:

```

class Binary_Mul_Padic_rep  $\supseteq$  Padic_rep $p$ 
   $a, b$ : Padic $p$ 
   $\beta^a, \beta^b, \delta^a, \delta^b, \gamma$ : vectors over  $\mathbb{Z}$ , with indices starting from 0.

  constructor ( $\tilde{a}$ : Padic $p$ ,  $\tilde{b}$ : Padic $p$ )
     $a := \tilde{a}, b := \tilde{b}$ 
    Initialize  $\beta^a, \beta^b, \delta^a, \delta^b$  and  $\gamma$  with empty vectors

  method next()
    If  $n + 2$  is a power of 2, then
      Resize  $\beta^a, \beta^b, \delta^a, \delta^b$  and  $\gamma$  to  $l_n + 1$ 
      Fill the new entries with zeros
     $\varepsilon^a := a_n, \varepsilon^b := b_n, \tau := 0$ 

    for  $q$  from 0 to  $l_n$  do
      if  $q > 0$  then
         $\varepsilon^a := \beta_{q-1}^a + p^{2^{q-1}} \varepsilon^a$ 
         $\varepsilon^b := \beta_{q-1}^b + p^{2^{q-1}} \varepsilon^b$ 
      if  $n + 2 = 2^{q+1}$  then
         $\delta_q^a := \varepsilon^a, \delta_q^b := \varepsilon^b$ 
         $\tau += \delta_q^a \varepsilon^b + \gamma_q$ 

```

```

if  $n + 2 = 2^{q+1}$  then break
 $\tau += \varepsilon^a \delta_q^b$ 

 $\beta_{l_n}^a := \varepsilon^a, \beta_{l_n}^b := \varepsilon^b$ 

for  $q$  from  $l_n$  down to 0 do
   $\gamma_q := \text{quo}(\tau, p^{2^q}), \tau := \text{rem}(\tau, p^{2^q})$ 

return  $\tau$ 

```

**Proposition II.7.** *Given two relaxed  $p$ -adic integers  $a$  and  $b$ , the computation of the product  $ab$  up till precision  $n$  can be done using  $O(l(n \log p) \log n)$  bit-operations and  $O(n \log p)$  bit-space.*

**Proof.** When computing  $\varphi_n$ , the vectors  $\beta^a, \beta^b, \delta^a, \delta^b$  and  $\gamma$  are resized to  $r_n$ , where  $r_n$  is the largest integer such that  $2^{r_n} \leq n + 2$ . From  $2^{r_n+1} > n + 2 \geq 2^{l_n+1}$ , we deduce that  $r_n > l_n$ , which means that the read and write operations in these vectors are licit.

For any integers  $n$  and  $q$  such that  $2^{q+1} < n + 2$ , we write  $\mu(n, q)$  for the largest integer less than  $n$  such that  $l_{\mu(n, q)} = q$ . We shall prove by induction that, when entering next for computing  $\varphi_n$ , the following properties hold for all  $q \geq 0$  such that  $2^{q+1} < n + 2$ :

- $\beta_q^a = a_{(k-1)2^q-1 \dots k2^q-1} = a_{\mu(n, q)-2^q+1 \dots \mu(n, q)+1}$  where  $k = \frac{\mu(n, q)+2}{2^q}$ , and similarly for  $\beta_q^b$ ,
- $\delta_q^a = a_{2^q-1 \dots 2^{q+1}-1}$ , and similarly for  $\delta_q^b$ , and
- $\gamma_q \leq 2p^{2^q}$ .

These properties trivially hold for when  $n = 0$ . Let us assume that they hold for a certain  $n \geq 0$ .

Now we claim that, at the end of step  $q$  of the first loop, the value of  $\varepsilon_n^a$  is  $a_{(k-1)2^q-1 \dots k2^q-1} = a_{n-2^q+1 \dots n+1}$  with  $k = (n + 2)/2^q$ . This clearly holds for when  $q = 0$  because  $\varepsilon^a = a_n$  and  $k = n + 2$ . Now assume that this claim holds until step  $q - 1$  for some  $q \geq 1$ . When entering step  $q$ , we have that  $\mu(n, q - 1) = n - 2^{q-1}$ , and part (a) of the induction hypothesis gives us that  $\beta_{q-1}^a = a_{n-2^{q-1}+1 \dots n-2^{q-1}+1}$ . From these quantities, we deduce:

$$\begin{aligned} \beta_{q-1}^a + p^{2^{q-1}} \varepsilon^a &= a_{n-2^q+1 \dots n-2^{q-1}+1} + p^{2^{q-1}} a_{n-2^{q-1}+1 \dots n+1} \\ &= a_{n-2^q+1 \dots n+1} = a_{(k-1)2^q-1 \dots k2^q-1}, \end{aligned}$$

with  $k = (n + 2)/2^q$ , which concludes the claim by induction. If  $n + 2$  is not a power of 2 then part (a) is clearly ensured at the end of the computation of  $\varphi_n$ . Otherwise  $n + 2 = 2^{l_n+1}$ , and  $\beta_{l_n}^a$  is set to  $a_{n-2^{l_n}+1 \dots n+1}$ , and part (a) is again satisfied when entering the computation of  $\varphi_{n+1}$ .

When  $\delta_q^a$  is set to  $\varepsilon^a$ , the value of  $\varepsilon^a$  is  $a_{(k-1)2^q-1 \dots k2^q-1}$  with  $k = 2$ . This ensures that part (b) holds when entering the computation of  $\varphi_{n+1}$ .

As to (c), during step  $q$  of the first loop, the value of  $\tau$  is incremented by at most

$$2(p^{2^q} - 1)^2 + 2p^{2^q} \leq 2p^{2^{q+1}} - 2p^{2^q} + 2.$$

At the end of this loop, we thus have

$$\tau \leq 2 p^{2^{l_n+1}} - 2 p + 2 (l_n + 1).$$

It follows that  $\tau/p^{2^{l_n}} < 2 p^{2^{l_n}} + 2 (l_n + 1)/p^{2^{l_n}}$ . If  $l_n \in \{0, 1\}$  then it is clear that  $2 (l_n + 1) \leq p^{2^{l_n}}$ , since  $p \geq 2$ . If  $l_n \geq 1$  then  $\frac{d(p^{2^{l_n}})}{dl_n} = \log(2) \log(p) 2^{l_n} p^{2^{l_n}} \geq 8 (\log 2)^2 \geq 2$ . We deduce that  $2 (l_n + 1) \leq p^{2^{l_n}}$  holds for all integer  $l_n \geq 0$ . Before exiting the function we therefore have that  $\gamma_{l_n} \leq 2 p^{2^{l_n}}$ ,  $\gamma_{l_n-1} \leq p^{2^{l_n-1}} \leq 2 p^{2^{l_n-1}}$ , etc., which completes the induction.

Since  $n + 2 = 2^{l_n} u$ , with  $u \geq 2$ , we have  $n + 2^k + 2 = 2^k (2^{l_n-k} u + 1)$ , whence  $l_{n+2^k} \geq k$ , for any  $k \leq l_n$ . All the carries stored in  $\gamma$  are therefore properly taken into account. This proves the correctness of the algorithm.

At precision  $n$ , summing the costs of all the calls to `next` amounts to

$$\begin{aligned} O\left(\sum_{i=0}^n \sum_{q=0}^{l_i} l(2^q \log p)\right) &= O\left(\sum_{2^q \leq n} \frac{n}{2^q} l(2^q \log p)\right) \\ &= O\left(l(n \log p) \sum_{2^q \leq n} 1\right) \\ &= O(l(n \log p) \log n). \end{aligned}$$

Furthermore,

$$O\left(\sum_{2^{q+1} \leq n+2} 2^q \log p\right) = O(n \log p)$$

provides a bound for the total bit-size of the auxiliary vectors  $\beta^a$ ,  $\beta^b$ ,  $\delta^a$ ,  $\delta^b$  and  $\gamma$ .  $\square$

Again, in practice, one should double the allocated sizes of the auxiliary vectors each time needed so that the cost of the related memory allocations and copies becomes negligible. In addition, for efficiency, one should precompute the powers of  $p$ .

### 3.3 Timings

In following Table II.2, we compare timings for power series over  $\mathbb{F}_p$ , and for  $p$ -adic integers *via* the technique of Section 2.7, called `Series_Mul_Padic_rep_p`, and *via* `Binary_Mul_Padic_rep_p` of Proposition II.7. In `Series_Mul_Padic_rep_p` the internal series product is the relaxed one reported in the first line.

$n$	8	16	32	64	128	256	512	1024	2048	4096
Relaxed mul. in $\mathbb{F}_p[[x]]$	2	7	20	50	140	360	930	2300	5700	14000
<code>Series_Mul_Padic_p</code>	19	59	160	420	1100	2600	6200	14000	34000	79000
<code>Binary_Mul_Padic_p</code>	8	16	37	89	170	360	800	1900	4300	10000
<code>Naive_Mul_Padic_p</code>	2.9	3.8	8.3	22	68	250	920	3600	14000	56000

**Table II.2.** Fast relaxed products, and naive lazy product, for  $p = 536870923$ , in microseconds.

For convenience, we recall the timings for the naive algorithm of Section 2.6 in the last line of Table II.2. We see that our `Binary_Mul_Padic_rep_p` is faster from size 512 on. Since handling small numbers with GMP is expensive, we also observe some overhead for small sizes, compared to the fast relaxed product of formal power series. On the other hand, since the relaxed product for power series makes internal use of Kronecker substitution, it involves integers that are twice as large as those in `Binary_Mul_Padic_rep_p`. Notice finally that the lifting strategy `Series_Mul_Padic_rep_p`, described in Section 2.7, is easy to implement, but not competitive.

## 4 Blockwise product

As detailed in Section 4.1 below for  $R = \mathbb{Z}$ , the relaxed arithmetic is slower than direct computations modulo  $p^n$  in binary representation. In [Hoe07b], an alternative approach for relaxed power series multiplication was proposed, which relies on grouping blocks of  $k$  coefficients and reducing a relaxed multiplication at order  $n$  to a relaxed multiplication at order  $n/k$ , with FFT-ed coefficients in  $M^{2k-1}$ .

Unfortunately, we expect that direct application of this strategy to our case gives rise to a large overhead. Instead, we will now introduce a variant, where the blocks of size  $k$  are rather rewritten into an integer modulo  $p^k$ . This aims at decreasing the overhead involved by the control instructions when handling objects of small sizes, and also improving the performance in terms of memory management by choosing blocks well suited to the sizes of the successive cache levels of the platform being used.

We shall start with comparisons between the relaxed and zealous approaches. Then we develop a supplementary strategy for a continuous transition between the zealous and the relaxed models.

### 4.1 Relaxed versus zealous

The first line of Table II.3 below displays the time needed for the product modulo  $p^n$  of two integers taken at random in the range  $0, \dots, p^n - 1$ . The next line concerns the performance of our function `binary` that converts a finite  $p$ -expansion of size  $n$  into its binary representation. The reverse function, reported in the last line, and written `expansion`, takes an integer in  $0, \dots, p^n - 1$  in base 2 and returns its  $p$ -adic expansion.

$n$	8	16	32	64	128	256	512	1024	2048	4096	8192
mod. mul.	0.38	0.52	1.0	2.9	9.0	27	85	250	690	1800	4500
binary	1.0	2.2	4.5	9.8	20	44	100	250	560	1400	3300
expansion	2.5	5.2	10	22	46	96	220	490	1200	3000	7300

**Table II.3.** Zealous product modulo  $p^n$  and conversions, for  $p = 536870923$ , in microseconds.

Let us briefly recall that binary can be computed fast by applying the classical divide and conquer paradigm as follows:

$$\text{binary}(a, p^n) = \text{binary}(a_{0\dots h}, p^h) + p^h \text{binary}(a_{h\dots n}, p^{n-h}), \text{ where } h = \lfloor n/2 \rfloor,$$

which yields a cost in  $O(l(n \log p) \log n)$ . Likewise, the same complexity bound holds for expansion. Within our implementation we have observed that these asymptotically fast algorithms outperform the naive ones whenever  $p^n$  is more than around 32 machine words.

Compared to Tables II.1 and II.2, these timings confirm the theoretical bounds: the relaxed product does not compete with a direct modular computation in binary representation. This is partly due to the extra  $O(\log n)$  factor for large sizes. But another reason is the overhead involved by the use of GMP routines with integers of a few words. In Table II.4, we report on the naive and the relaxed products in base  $p^{32}$ . Now we see that our naive product becomes of the same order of efficiency as the zealous approach up to precision 1024. The relaxed approach starts to win when the precision reaches 256 in base  $p^{32}$ .

$kl$	32	64	128	256	512	1024	2048	4096	8192
Naive_Mul_Padic <sub><math>p^k</math></sub>	1.8	4.1	10	27	84	280	1000	3900	15000
Binary_Mul_Padic <sub><math>p^k</math></sub>	3.2	6.1	16	53	170	570	1800	5300	15000

**Table II.4.** Relaxed product modulo  $(p^k)^l$ , for  $p = 536870923$  and  $k = 32$ , in microseconds.

## 4.2 Monoblock strategy

If one wants to compute the product  $ab$  of two  $p$ -adic numbers  $a$  and  $b$ , then: one can start by converting both of them into  $p^k$ -adic numbers  $A$  and  $B$  respectively, multiply  $A$  and  $B$  as  $p^k$ -adic numbers, and finally convert  $AB$  back into a  $p$ -adic number. The transformations between  $p$ -adic and  $p^k$ -adic numbers can be easily implemented:

```

class To_Blocks $p^k$   $\supseteq$  Padic_rep $p^k$ 
  a: Padic $p$ 
  constructor ( $\tilde{a}$ : Padic $p$ )
    a :=  $\tilde{a}$ 
  method next()
    return binary( $a_{nk\dots(n+1)k}$ )

class From_Blocks $p$   $\supseteq$  Padic_rep $p$ 
  a: Padic $p^k$ 
  b:  $p$ -expansion of size  $k$ 
  constructor ( $\tilde{a}$ : Padic $p^k$ )
    a :=  $\tilde{a}$ 
  method next ()
    if  $n \bmod k = 0$  then  $b = p\_expansion(a_{n/k}, p)$ 
    return  $b_{n \bmod k}$ 

```

If `to_blocks` and `from_blocks` represent the top level functions then the product  $c$  of  $a$  and  $b$  can be simply obtained as  $c = \text{from\_blocks}(\text{to\_blocks}(a) \text{ to\_blocks}(b))$ . We call this way of computing products the *monoblock strategy*.

Notice that choosing  $k$  very large is similar to zealous computations. This monoblock strategy can thus be seen as a mix of the zealous and the relaxed approaches. However, it is only relaxed for  $p^k$ -expansions, not for  $p$ -expansions. In fact, let  $A$  and  $B$  still denote the respective  $p^k$ -adic representations of  $a$  and  $b$ , so that  $c = \text{from\_blocks}(C)$ , for  $C = A B$ . Then the computation of  $c_0$  requires the knowledge of  $C_0 = A_0 B_0$ , whence it depends on all the coefficients  $a_0, \dots, a_{k-1}$  and  $b_0, \dots, b_{k-1}$ , which breaks the main requirement on relaxed operations recalled in Section 1.1.1. In the next paragraphs we derive an actual relaxed product from this strategy, at the price of a reasonable overhead.

### 4.3 Relaxed blockwise product

We are now to present a relaxed  $p$ -adic blockwise product. This product depends on two integer parameters  $m$  and  $k$ . The latter still stands for the size of the blocks to be used, while the former is a threshold: below precision  $m$  one calls a given product on  $p$ -expansions, while in large precision an other product is used on  $p^k$ -expansions.

If  $a$  and  $b$  are the two numbers in  $R_p$  that we want to multiply as  $p$ -expansions, then we first rewrite them  $a = a_{0\dots m} + p^m \bar{a}$  and  $b = b_{0\dots m} + p^m \bar{b}$ , where

$$\bar{a} = a/p^m = \sum_{i=0}^{\infty} a_{m+i} p^i \quad \text{and} \quad \bar{b} = b/p^m = \sum_{i=0}^{\infty} b_{m+i} p^i.$$

Now multiplying  $a$  and  $b$  gives

$$c = a_{0\dots m} b_{0\dots m} + p^m (a_{0\dots m} \bar{b} + \bar{a} b_{0\dots m}) + p^{2m} \bar{a} \bar{b},$$

where the product  $\bar{a} \bar{b}$  can be computed in base  $p^k$ , as it is detailed in the following implementation:

```

class Blocks_Mul_Padic_rep_p  $\triangleright$  Padic_rep_p
  a, b, c,  $\bar{a}$ ,  $\bar{b}$ : Padic_p
   $\bar{A}$ ,  $\bar{B}$ : Padic_{p^k}
  constructor ( $\bar{a}$ : Padic_p,  $\bar{b}$ : Padic_p)
    a :=  $\bar{a}$ , b :=  $\bar{b}$ 
     $\bar{a}$  := a/p^m,  $\bar{b}$  := b/p^m
     $\bar{A}$  := to_blocks( $\bar{a}$ ),  $\bar{B}$  := to_blocks( $\bar{b}$ )
     $\bar{c}$  := from_blocks( $\bar{A} \bar{B}$ )
    c := a_{0\dots m} b_{0\dots m} + p^m (a_{0\dots m}  $\bar{b}$  +  $\bar{a}$  b_{0\dots m}) + p^{2m}  $\bar{c}$ 
  method next()
  return c_n

```

In Figure II.2 below, we illustrate the contribution of each  $a_i b_j$  to the product  $c_{i+j}$  computed with the present blockwise version. In both bases  $p$  and  $p^k$  the naive product is used, and the numbers inside the squares indicate the degrees at which the corresponding product is actually computed.

10	11	12												
9	10	11	10						14					
8	9	10												
7	8	9												
6	7	8	6						10					
5	6	7												
4	5	6												
3	4	5												
2	3	4	5	6	7	8	9	10	11	12				
1	2	3	4	5	6	7	8	9	10	11				
0	1	2	3	4	5	6	7	8	9	10				

Figure II.2. Blockwise product for  $m = 3$  and  $k = 4$ .

**Proposition II.8.** *If  $m \geq k - 1$ , then  $\text{Blocks\_Mul\_Padic\_rep}_{p^k}$  is relaxed for base  $p$ .*

**Proof.** It is sufficient to show that the computation of  $\bar{c}_{n-2m}$  only involves terms in  $a$  and  $b$  of degree at most  $n$ . In fact  $\bar{c}_{n-2m}$  requires the knowledge of the coefficients of  $\bar{A}$  and  $\bar{B}$  to degree at most  $l = \lfloor (n - 2m)/k \rfloor$ , hence the knowledge of the coefficients of  $a$  and  $b$  to degree  $k(l + 1) - 1 + m \leq n - 2m + k - 1 + m = n + k - 1 - m$ , which concludes the proof thanks to the assumption on  $m$ . Notice that the latter inequality is an equality whenever  $n - 2m$  is a multiple of  $k$ . Therefore  $m \geq k - 1$  is necessary to ensure the product to be relaxed.  $\square$

#### 4.4 Timings

In the following table, we use blocks of size  $k = 32$ , and compare the blockwise versions of the naive product of Section 2.6 to the relaxed one of Section 3.2. The first line concerns the monoblock strategy: below precision 32 we directly use the naive  $p$ -adic product; for larger precisions we use the naive  $p^k$ -adic product. The second line is the same as the first one except that we use a relaxed  $p^k$ -adic product. In the third line the relaxed blockwise version is used with  $m = 32$ : we use the naive product for both  $p$ - and  $p^k$ -adic expansions. The fourth line is similar except that the fast relaxed product is used beyond precision 32.

$n$	8	16	32	64	128	256	512	1024	2048	4096
mono Naive_Mul_Padic <sub><math>p</math></sub>	3.5	5.5	11	53	95	190	380	870	2200	6500
mono Binary_Mul_Padic <sub><math>p</math></sub>	3.4	5.5	11	57	110	220	500	1200	3000	7800
blocks Naive_Mul_Padic_rep <sub><math>p</math></sub>	8.0	11	17	46	140	320	700	1600	3700	9400
blocks Binary_Mul_Padic_rep <sub><math>p</math></sub>	8.0	11	17	46	140	330	750	1700	3900	9100

Table II.5. Blockwise products to precision  $n$ , for  $p = 536870923$  and  $k = 32$ , in microseconds.

When compared to Table II.4, we can see that most of the time within the monoblock strategy is actually spent on base conversions. In fact, the strategy does not bring a significant speed-up for a single product, but for more complex computations, the base conversions can often be factored.

For instance, assume that  $a$  and  $b$  are two  $d \times d$  matrices with entries in  $\mathbb{Z}_p$ . Then the multiplication  $c = a b$  involves only  $O(d^2)$  base conversions and  $O(d^3)$   $p^k$ -adic products. For large  $d$ , the conversions thus become inexpensive. In Section 7, we will encounter a similar application to multiple root extraction.

## 5 Application to recursive $p$ -adic numbers

A major motivation behind the relaxed computational model is the efficient expansion of  $p$ -adic numbers that are solutions to recursive equations. This section is an extension of [Hoe02, Section 4.5] to  $p$ -adic numbers.

Let us slightly generalize the notion of a recursive equation, which was first defined in the introduction, so as to accommodate for initial conditions. Consider a functional equation

$$Y = \Phi(Y), \quad (\text{II.2})$$

where  $Y$  is a vector of  $d$  unknowns in  $R_p$ . Assume that there exist a  $k \in \mathbb{N}^*$  and *initial conditions*  $c_0, \dots, c_{k-1} \in M^d$ , such that for all  $n \geq k$  and  $y, \tilde{y} \in R_p^d$  with  $y_0 = c_0, \dots, y_{k-1} = c_{k-1}$ , we have

$$(\tilde{y} - y) \in p^n R_p^d \implies (\Phi(\tilde{y}) - \Phi(y)) \in p^{n+1} R_p^d. \quad (\text{II.3})$$

Stated otherwise, this condition means that each coefficient  $b_n = \Phi(b)_n$  with  $n \geq k$  only depends on previous coefficients  $b_0, \dots, b_{n-1}$ . Therefore, setting  $c = c_0 + c_1 p + \dots + c_{k-1} p^{k-1}$ , the sequence  $c, \Phi(c), \Phi(\Phi(c))$  converges to a unique solution  $b \in R_p^d$  of (II.2) with  $b_0 = c_0, \dots, b_{k-1} = c_{k-1}$ . We will call (II.2) a *recursive equation* and the entries of the solution  $b$  *recursive  $p$ -adic numbers*.

### 5.1 Implementation

Since  $p$  induces an element  $\mathbf{p} = (p, \dots, p)$  in  $R^d$  and an isomorphism  $R_p^d \simeq (R^d)_{\mathbf{p}}$ , we may reinterpret a solution  $b = \Phi(b)$  as a  $\mathbf{p}$ -adic number over  $R^d$ . Using this trick, we may assume without loss of generality that  $d = 1$ . In our implementation, recursive numbers are instances of the following class that stores the initial conditions  $b_0, \dots, b_{k-1}$  and the equation  $\Phi$ :

```

class Recursive_Padic_rep  $\triangleright$  Padic_rep_p
   $\Phi$ : function from  $R_p$  to  $R_p$ 
   $b_0, \dots, b_{k-1}$ : initial conditions in  $M$ 
  constructor ( $\tilde{\Phi}$ : function,  $\tilde{b}_0, \dots, \tilde{b}_{k-1}$ :  $M$ )
     $\Phi := \tilde{\Phi}$ ,  $b_0 := \tilde{b}_0, \dots, b_{k-1} := \tilde{b}_{k-1}$ 
  method next()
    If  $n < k$  then return  $b_n$ 

```

**return**  $\Phi(\text{this})_n$

In the last line, the expression  $\Phi(\text{this})$  means the evaluation of  $\Phi$  at the concrete instance of the  $p$ -adic  $b = \Phi(b)$  being currently defined.

**Example II.9.** Consider  $\Phi(b) = pb + 1$ , with one initial condition  $b_0 = 1$ . It is clear that  $b$  is recursive, since the  $n$  first terms of  $pb$  can be computed from the only  $n - 1$  first terms of  $b$ . We have  $b_1 = b_2 = \dots = 1$ . In fact,  $b = 1/(1 - p)$ .

## 5.2 Complexity analysis

If  $\Phi$  is an expression built from  $L$  constants, sums, differences, and products (all of arity two), then the computation of  $b$  simply consists of performing these  $L$  operations in the relaxed model. For instance, when using the relaxed product of Proposition II.6, this amounts to  $O(L \log_p(n) \log n)$  operations to obtain the  $n$  first terms of  $b$ .

This complexity bound is to be compared to the classical approach *via* the Newton operator. In fact, one can compute  $b$  with fixed-point  $p$ -adic arithmetic by evaluating the following operator  $N_\Phi(z) = z - (z - \Phi(z))/(1 - \Phi'(z))$ . There are several cases where the relaxed approach is faster than the Newton operator:

1. The constant hidden behind the “ $O$ ” of the Newton iteration is higher than the one with the relaxed approach. For instance, if  $b$  is really a vector in  $R_p^d$ , then the Newton operator involves the inversion of a  $d \times d$  matrix at precision  $n/2$ , which gives rise to a factor  $O(d^3)$  in the complexity (assuming the naive matrix product is used). The total cost of the Newton operator to precision  $n$  in  $\mathbb{Z}_p$  is thus in  $O((dL + d^3) \log_p(n))$ . Here  $O(dL)$  bounds the number of operations needed to evaluate the Jacobian matrix. In this situation, if  $L \ll d^2$ , and unless  $n$  is very large, the relaxed approach is faster. This will be actually illustrated in the next subsection.
2. Even in the case  $d = 1$ , the “ $O$ ” hides a non trivial constant factor due to a certain amount of “recomputations”. For moderate sizes, when polynomial multiplication uses Karatsuba’s algorithm, or the Toom-Cook method, the cost of relaxed multiplication also drops to a constant times the cost of zealous multiplication [Hoe02, Hoe07b]. In such cases, the relaxed method often becomes more efficient. This will be illustrated in Section 6 for the division.
3. When using the blockwise method from Section 4 or [Hoe07b] for power series, the overhead of relaxed multiplication can often be further reduced. In practice, we could observe that this makes it possible to outperform Newton’s method even for very large sizes.

For more general functional equations, where  $\Phi$  involves non-algebraic operations, it should also be noticed that suitable Newton operators  $\Phi$  are not necessarily available. For instance, if the mere definition of  $\Phi$  involves  $p$ -expansions, then the Newton operator may be not defined anymore, or one needs to explicitly compute with  $p$ -expansions. This occurs for instance for  $R = \mathbb{Z}$ , when  $\Phi$  involves the “symbolic derivation”  $b \mapsto \sum_{n \geq 1} n b_n p^{n-1}$ .

### 5.3 Timings

In order to illustrate the performance of the relaxed model with respect to Newton iteration, we consider the following family of systems of  $p$ -adic integers:

$$\Phi_{d,i}(x_1, \dots, x_d) = 1 + p \sum_{k=1}^d (k+i) x_k^{(k+i) \bmod 3}, \text{ for } i \in \{1, \dots, d\}.$$

The number of  $p$ -adic products grows linearly with  $d$ . Yet, the total number of operations grows with  $d^2$ .

In Table II.6, we compute the 256 first terms of the solution  $b = \Phi_d(b)$  with the initial condition  $b = (1, \dots, 1) + O(p)$ . We use the naive product of Section 2.6 and compare to the Newton iteration directly implemented on the top of the routines of GMP. In fact the time we provide in the line ‘‘Matrix multiplication’’ does not corresponds to a complete implementation of the iteration but only to two products of two  $d \times d$  matrices with integers modulo  $p^{64}$ . These two operations necessarily occurs for inverting the Jacobian matrix to precision  $p^{128}$  when using the classical algorithm as described in [GG03, Algorithm 9.2]. This can be seen as a lower bound for any implementation of the Newton method. However the line ‘‘Newton implementation’’ corresponds to our implementation of this method, hence this is an upper bound. The next line of the table, named ‘‘Naive iteration’’, corresponds to the computation from  $b = (1, \dots, 1)$  of  $\Phi_d(b)$  modulo  $p^2$ , then  $\Phi_d(\Phi_d(b))$  modulo  $p^3$ , etc. This sequence converges linearly to the solution.

$d$	1	2	4	8	16	32	64	128
Matrix multiplication	0.002	0.014	0.12	0.9	7.2	56	460	3600
Newton implementation	0.13	0.31	2.3	13	95	700	5500	43000
Naive iteration	3.8	7.7	17	40	110	328	1114	3992
Naive_Mul_Padic <sub><math>p</math></sub>	0.34	0.42	1.4	3.3	8.7	26	94	420

**Table II.6.** 256 first terms of  $b = \Phi_d(b)$ , for  $p = 536870923$ , in milliseconds.

Although Newton iteration is faster for tiny dimensions  $d \leq 2$ , its cost grows as  $d^3$  for larger  $d$ , whereas the relaxed approach reported on the last line only grows as  $d^2$ . For  $d = 1$ , we notice that the number  $b$  is computed with essentially one relaxed product. Notice that, due to the linear convergence, the naive iteration behaves well when the dimension is large and the precision relatively small.

In the next table we report of the same computations but with the relaxed product of Section 3.2 at precision 1024; the conclusions are essentially the same:

$d$	1	2	4	8	16	32	64	128
Matrix multiplication	0.014	0.12	0.98	7.9	62	490	4000	31000
Newton implementation	0.58	1.2	13	90	640	4900	38000	300000
Naive iteration	109	217	446	934	2047	4806	12537	36542
Binary_Mul_Padic <sub><math>p</math></sub>	2.4	2.6	9.4	21	52	150	480	2300

**Table II.7.** 1024 first terms of  $b = \Phi_d(b)$ , for  $p = 536870923$ , in milliseconds.

## 6 Relaxed division

We are now to present relaxed algorithms to compute the quotient of two  $p$ -adic numbers. The technique is similar to power series, as treated in [Hoe02], but with subtleties.

### 6.1 Division by a “scalar”

The division of a power series in  $\mathbb{K}[[x]]$  by an element of  $\mathbb{K}$  is immediate, but it does not extend to  $p$ -adic numbers, because of the propagation of the carries. We shall introduce two new operations. Let  $\beta \in M$  play the role of a “scalar”. The first new operation, written  $\text{mul\_rem}(\beta: M, a: \text{Padic}_p)$ , returns the  $p$ -adic number  $c$  with coefficients  $c_n = \text{rem}(\beta a_n, p)$ . The second operation, written  $\text{mul\_quo}(\beta: M, a: \text{Padic}_p)$ , returns the corresponding carry, so that

$$\begin{aligned} \beta a &= \text{mul\_rem}(\beta, a) + p \text{mul\_quo}(\beta, a) \\ &= \sum_{n=0}^{\infty} \text{rem}(\beta a_n, p) p^n + \sum_{n=0}^{\infty} \text{quo}(\beta a_n, p) p^{n+1}. \end{aligned}$$

These operations are easy to implement, as follows:

```

class Mul_Rem_Padic_rep_p ⊇ Padic_rep_p
  β: M
  a: Padic_p
  constructor (β̃: M, ā: Padic_p)
    β := β̃, a := ā
  method next()
    return rem(β a_n, p)

class Mul_Quo_Padic_rep_p ⊇ Padic_rep_p
  β: M
  a: Padic_p
  constructor (β̃: M, ā: Padic_p)
    β := β̃, a := ā
  method next()
    return quo(β a_n, p)

```

**Proposition II.10.** *Let  $a$  be a relaxed  $p$ -adic number and let  $\beta \in M$ . If  $\beta$  is invertible modulo  $p$ , with given inverse  $\gamma = \beta^{-1} \bmod p$ , then the quotient  $c = a/\beta$  is recursive and  $c$  satisfies the equation*

$$c = \text{mul\_rem}(\gamma, a - p \text{mul\_quo}(\beta, c)), \quad c_0 = \gamma a_0 \bmod p.$$

If  $R = \mathbb{Z}$ , then  $a/\beta$  can be computed up till precision  $n$  using  $O(n \log p)$  bit-operations.

**Proof.** It is clear from the definitions that the proposed formula actually defines a recursive number. Then, from  $\beta c = \text{mul\_rem}(\beta, c) + p \text{mul\_quo}(\beta, c)$ , we deduce that  $\beta c - p \text{mul\_quo}(\beta, c) = \text{mul\_rem}(\beta, c)$ , hence

$$c = \text{mul\_rem}(\gamma, \beta c - p \text{mul\_quo}(\beta, c)) = \text{mul\_rem}(\gamma, a - p \text{mul\_quo}(\beta, c)).$$

The functions `mul_rem` and `mul_quo` both take  $O(n \lceil \log p \rceil)$  bit-operations if  $R = \mathbb{Z}$ , which concludes the proof.  $\square$

## 6.2 Quotient of two $p$ -adic numbers

Once the division by a “scalar” is available, we can apply a similar formula as for the division of power series of [Hoe02].

**Proposition II.11.** *Let  $a$  and  $b$  be two relaxed  $p$ -adic numbers such that  $b_0$  is invertible of given inverse  $\gamma = b_0^{-1} \bmod p$ . The quotient  $c = a/b$  is recursive and satisfies the following equation:*

$$c = \frac{a - (b - b_0)c}{b_0}, \quad c_0 = \gamma a_0 \bmod p.$$

If  $R = \mathbb{Z}$ , then  $a/b$  can be computed up till precision  $n$  using  $O(\lceil n \log p \rceil \log n)$  bit-operations.

**Proof.** The last assertion on the cost follows from Proposition II.7.  $\square$

**Remark II.12.** Notice that  $p$  is not assumed to be prime, so that we can replace  $p$  by  $p^k$ , and thus benefit from the monoblock strategy of Section 4.2. This does not involve a large amount of work: it suffices to write `from_blocks(to_blocks(a)/to_blocks(b))`. Notice that this involves inverting  $b_{0\dots p^k}$  modulo  $p^k$ .

## 6.3 Timings

In following Table II.8 we display the computation time for our division algorithm. We compare several methods:

- The first line “Newton” corresponds to the classical Newton iteration [GG03, Algorithm 9.2] used in the zealous model.
- The second line corresponds to one call of GMP’s extended g.c.d. function.
- The third line is a comparison with PARI/GP version 2.3.5.
- The next two lines `Naive_Mul_Padicp` and `Binary_Mul_Padicp` correspond to the naive product of Section 2.6, and the relaxed one of Section 3.2.
- Then the next lines “mono `Naive_Mul_Padicp`” and “mono `Naive_Mul_Padicp`” correspond to the monoblock strategy from Section 4.2 with blocks of size 32.
- Similarly the lines “blocks `Naive_Mul_Padicp`” and “blocks `Naive_Mul_Padicp`” correspond to the relaxed block strategy from Section 4.3 with blocks of size 32.
- Finally the last line corresponds to direct computations in base  $p^{32}$  (with no conversions from/to base  $p$ ).

When compared to Tables II.1 and II.2, we observe that the cost of one division algorithm is merely that of one multiplication whenever the size becomes sufficiently large, as expected. We also observe that our “monoblock division” is faster than the zealous one for large sizes; this is even more true if we directly compute in base  $p^{32}$ .

$n$	8	16	32	64	128	256	512	1024	2048
Newton	3	4	7	18	49	140	430	1300	3700
GMP's extended g.c.d.	3	6	14	35	92	250	730	2200	5600
PARI/GP	0.68	1.1	2.8	8.5	28	99	360	1300	4800
Naive_Mul_Padic $_p$	4	7	15	35	95	300	1000	3700	14000
Binary_Mul_Padic $_p$	9	21	44	93	200	420	920	2000	4800
mono Naive_Mul_Padic $_p$	6	10	20	95	160	280	540	1200	2800
mono Binary_Mul_Padic $_p$	6	10	20	110	170	320	660	1500	3600
blocks Naive_Mul_Padic $_p$	10	16	27	70	190	430	900	1900	4500
blocks Binary_Mul_Padic $_p$	10	16	27	65	180	410	900	2000	4500
Naive_Mul_Padic $_{p^{32}}$			6	22	42	88	200	500	1500

**Table II.8.** Divisions, for  $p = 536870923$ , in microseconds.

## 7 Higher order roots

For power series in characteristic 0, the  $r$ th root  $g$  of  $f$  is recursive, with equation  $g = \int f'/(r g^{r-1})$  and initial condition  $g_0 = f_0^{1/r}$  (see [Hoe02, Section 3.2.5] for details). This expression neither holds in small positive characteristic, nor for  $p$ -adic integers. In this section we propose new formulas for these two cases, which are compatible with the monoblock strategy of Section 4.2.

### 7.1 Regular case

In this subsection we treat the case when  $r$  is invertible modulo  $p$ .

**Proposition II.13.** *Assume that  $r$  is invertible modulo  $p$ , and let  $a$  be a relaxed invertible  $p$ -adic number in  $R_p$  such that  $a_0$  is an  $r$ th power modulo  $p$ . Then any  $r$ th root  $b_0$  of  $a_0$  modulo  $p$  can be uniquely lifted into an  $r$ th root  $b$  of  $a$ . Moreover,  $b$  is a recursive number for the equation*

$$b = \frac{a - b^r + r b_0^{r-1} b}{r b_0^{r-1}}. \quad (\text{II.4})$$

The  $n$  first terms of  $b$  can be computed using

- $O(\log r M(n) \log n)$  operations in  $\mathbb{K}$ , if  $R = \mathbb{K}[[x]]$  and  $p = x$ , or
- $O(\log r \mid (n \log p) \log n)$  bit-operations, if  $R = \mathbb{Z}$ .

**Proof.** Since  $r$  is invertible modulo  $p$ , the polynomial  $x^r - a$  is separable modulo  $p$ . Any of its roots modulo  $p$  can be uniquely lifted into a root in  $R_p$  by means of the classical Newton operator [Lan02, Proposition 7.2].

Since  $a_0$  is invertible, so is  $b_0$ . It is therefore clear that equation (II.4) uniquely defines  $b$ , but it is not immediately clear how to evaluate it so that it defines a recursive number. For this purpose we rewrite  $b$  into  $b_0 + c$ , with  $c$  of valuation at least 1:

$$b = \frac{a - (b_0 + c)^r + r b_0^{r-1} (b_0 + c)}{r b_0^{r-1}} = \frac{a - \sum_{k=2}^r \binom{r}{k} b_0^{r-k} c^k + (r-1) b_0^r}{r b_0^{r-1}}.$$

Since  $r$  is invertible modulo  $p$ , we now see that it does suffice to know the terms to degree  $n-1$  of  $b$  in order to deduce  $b_n$ .

The latter expanded formula is suitable for an implementation but unfortunately the number of products to be performed grows linearly with  $r$ . Instead we modify the classical binary powering algorithm to compute the expression needed with  $O(\log r)$  products only, as follows. In fact we aim at computing  $\beta_r = (b_0 + c)^r - r b_0^{r-1} c - b_0^r$  in a way to preserve the recursiveness. We proceed by induction on  $r$ .

If  $r=1$ , then  $\beta_r=0$ . If  $r=2$  then  $\beta_2=c^2$ . Assume that  $r=2h$ , and that  $\beta_h$  is available by induction. From

$$\begin{aligned} \beta_h^2 &= (b_0 + c)^r + (h b_0^{h-1} c + b_0^h)^2 - 2 (h b_0^{h-1} c + b_0^h) (\beta_h + h b_0^{h-1} c + b_0^h) \\ &= (b_0 + c)^r - (h b_0^{h-1} c + b_0^h)^2 - 2 (h b_0^{h-1} c + b_0^h) \beta_h, \end{aligned}$$

we deduce that

$$\begin{aligned} \beta_r &= \beta_h^2 + (h b_0^{h-1} c + b_0^h)^2 + 2 (h b_0^{h-1} c + b_0^h) \beta_h - r b_0^{r-1} c - b_0^r \\ &= \beta_h (\beta_h + 2 (h b_0^{h-1} c + b_0^h)) + (h b_0^{h-1} c)^2. \end{aligned}$$

Since  $\beta_h$  and  $c$  have positive valuation, the recursiveness is well preserved through this intermediate expression.

On the other hand, if  $r$  is odd then we can write  $r=h+1$ , with  $h$  even, and assume that  $\beta_h$  is available by induction. Then we have that:

$$\begin{aligned} \beta_r &= (b_0 + c) \beta_h + (b_0 + c) (h b_0^{h-1} c + b_0^h) - (h+1) b_0^h c - b_0^{h+1} \\ &= (b_0 + c) \beta_h + h b_0^{h-1} c^2. \end{aligned}$$

Again the recursiveness is well preserved through this intermediate expression. The equation of  $b$  can finally be evaluated using  $O(\log r)$  products and one division. By [Hoe02, Section 4.3.2, Theorem 6], this concludes the proof for power series. By Propositions II.7 and II.11, we also obtain the desired result for  $p$ -adic integers.  $\square$

For the computation of the  $r$ th root in  $\mathbb{Z}/p\mathbb{Z}$ , we have implemented the algorithms of [GG03, Theorems 14.4 and 14.9]: each extraction can be done with  $\tilde{O}(r \log p)$  bit-operations in average, with a randomized algorithm. This is not the bottleneck for our purpose, so we will not discuss this aspect longer in this chapter.

**Remark II.14.** Notice that  $(p)$  is not assumed to be prime in Proposition II.13. Therefore, if we actually have an  $r$ th root  $b$  of  $a$  modulo  $p^k$ , then  $b$  can be seen as a  $p^k$ -recursive number, still using equation (II.4). Hence, one can directly apply the monoblock strategy of Section 4.2 to perform internal computations modulo  $p^k$ .

## 7.2 $p$ th roots

If  $\mathbb{K}$  is a field of characteristic  $p$ , then  $f \in \mathbb{K}[[x]]$  is a  $p$ th power if, and only if,  $f \in \mathbb{K}^p[[x^p]]$ . If it exists, the  $p$ th root of a power series is unique. Here,  $\mathbb{K}^p$  represents the subfield of the  $p$ th powers of  $\mathbb{K}$ . By the way, let us mention that, for a general effective field  $\mathbb{K}$ , Fröhlich and Shepherdson have shown that testing if an element is a  $p$ th power is not decidable [FS56, Section 7] (see also the example in [Gat84, Remark 5.10]).

In general, for  $p$ -adic numbers, an  $r$ th root extraction can be almost as complicated as the factorization of a general polynomial in  $R_p[x]$ . For instance, with  $R = \mathbb{Z}[\sqrt{2}]$  and  $p = \sqrt{2}$  we have that  $r = 2 = p^2$  has valuation 2 in  $R_p$ . We will not cover such a general situation. We will only consider the case of the  $p$ -adic integers, that is for when  $R = \mathbb{Z}$  and  $p$  is prime.

From now on, let  $a$  denote a  $p$ -adic integer in  $\mathbb{Z}_p$  from which we want to extract the  $p$ th root (if it exists). If the valuation of  $a$  is not a multiple of  $p$ , then  $a$  is not a  $p$ th power. If it is a multiple of  $p$ , then we can factor out  $p^{\text{val } a}$  and assume that  $a$  has valuation 0. The following lemma is based on classical techniques, we briefly recall its proof for completeness:

**Lemma II.15.** *Assume that  $p$  is prime, and let  $a \in \mathbb{Z}_p$  be invertible.*

- *If  $p \geq 3$ , then  $a$  is a  $p$ th power if, and only if,  $a_0 + p a_1 = a_0^p$  modulo  $p^2$ . In this case there exists only one  $p$ th root.*
- *If  $p = 2$ , then  $a$  is a  $p$ th power if, and only if,  $a_1 = a_2 = 0$ . In this case there exist exactly two square roots.*

**Proof.** If  $a = b^p$  in  $\mathbb{Z}_p$  then  $b_0 = a_0$ . After the translation  $x = a_0 + y$  in  $x^p - a = 0$ , we focus on the equation  $(b_0 + y)^p - a = 0$ , which expands to

$$h(y) = y^p + \sum_{i=1}^{p-1} \binom{p}{i} b_0^{p-i} y^i - (a - b_0^p) = 0. \quad (\text{II.5})$$

For any  $i \in \{1, \dots, p-1\}$ , the coefficient  $\binom{p}{i}$  has valuation at least one because  $p$  is prime. Reducing the latter equation modulo  $p^2$ , it is thus necessary that  $a_0 + p a_1 = b_0^p$  modulo  $p^2$ .

Assume now that  $a_0 + p a_1 = b_0^p$  holds modulo  $p^2$ . After the change of variables  $y$  by  $p z$  and division by  $p^2$ , we obtain

$$\tilde{h}(z) = p^{p-2} z^p + \sum_{i=2}^{p-1} \binom{p}{i} b_0^{p-i} p^{i-2} z^i + b_0^{p-1} z - \frac{a - b_0^p}{p^2} = 0. \quad (\text{II.6})$$

We distinguish two cases:  $p \geq 3$  and  $p = 2$ .

If  $p \geq 3$ , then any root of  $\tilde{h}$  must be congruent to  $(a - b_0^p)/(b_0^{p-1} p^2)$  modulo  $p$ . Since  $\tilde{h}'((a - b_0^p)/(b_0^{p-1} p^2)) = b_0^{p-1}$  has valuation 0, the Newton operator again ensures that  $\tilde{h}$  has exactly one root [Lan02, Proposition 7.2].

If  $p = 2$ , then  $\tilde{h}(z)$  rewrites into  $z^2 + b_0 z - \frac{a - b_0^2}{p^2} = z^2 + z - \frac{a - 1}{4}$ . Since  $\tilde{h}'(z) = b_0 \pmod{p} = 1 \pmod{2}$ , any root of  $\tilde{h}$  modulo 2 can be lifted into a unique root of  $\tilde{h}$  in  $\mathbb{Z}_2$ . The possible roots being 0 and 1, this gives the extra condition  $a_2 = 0$ .  $\square$

### 7.3 Square roots in base 2

In the following proposition we show that the square root of a 2-adic integer can be expressed into a recursive number that can be computed with essentially one relaxed product.

**Proposition II.16.** *Let  $a$  be a relaxed 2-adic integer in  $\mathbb{Z}_2$  with  $a_0 = 1$  and  $a_1 = a_2 = 0$ . Let  $b$  be a square root of  $a$ , with  $b_0 = 1$  and  $b_1$  being 0 or 1, and let  $c = (b - b_0 - 2b_1)/4$ , and  $\tilde{a} = (a - (b_0 + 2b_1)^2)/8$ . Then  $c$  is a recursive number with initial condition  $c_0 = \tilde{a}_0$  and equation*

$$c = \frac{\tilde{a} - 2c^2}{b_0 + 2b_1}. \quad (\text{II.7})$$

The  $n$  first terms of  $b$  can be computed using  $O((n \log p) \log n)$  bit-operations.

**Proof.** Equation (II.7) simply follows from

$$(b_0 + 2b_1 + 4c)^2 - a = (b_0 + 2b_1)^2 + 8(b_0 + 2b_1)c + 16c^2 - a = 0.$$

The cost is a consequence of Propositions II.7 and II.11.  $\square$

**Remark II.17.** As in the preceding regular case, we can see  $c$  as a  $p^k$ -recursive number as soon as  $c$  is known modulo  $p^k$ . In fact letting  $C = C_0 + \tilde{C}$ , with  $C_0 = C \pmod{p^k}$ , equation (II.7) rewrites into

$$\begin{aligned} (b_0 + 2b_1)C &= \tilde{a} - 2(C_0 + (C - C_0))^2 \\ &= \tilde{a} - 2C_0^2 - 4C_0(C - C_0) - 2(C - C_0)^2, \end{aligned}$$

which gives

$$(b_0 + 2b_1 + 4C_0)C = \tilde{a} + 2C_0^2 - 2(C - C_0)^2.$$

The latter equation implies that  $C$  is  $p^k$ -recursive, so that we can naturally benefit from the monoblock strategy from Section 4.2.

### 7.4 $p$ th roots in base $p$

In this subsection we assume that  $p$  is an odd prime integer. We will show that the  $p$ th root is recursive and can be computed using similar but slightly more complicated formulas than in the regular case.

**Proposition II.18.** *Let  $a$  be an invertible relaxed  $p$ -adic integer in  $\mathbb{Z}_p$  such that  $a = a_0^p \pmod{p^2}$ . Let  $b$  denote the  $p$ th root of  $a$ , with  $b_0 = a_0$ , let  $\tilde{a} = (a - (b_0 + pb_1)^p)/p^2$ , and let  $c = (b - b_0)/p$ . Then  $c$  is a recursive number with initial condition  $c_0 = b_1$  and equation*

$$c = c_0 + \frac{\tilde{a} - \gamma_p}{(b_0 + pc_0)^{p-1}}, \quad (\text{II.8})$$

where

$$\gamma_r = \frac{(b_0 + p c)^r - r p (b_0 + p c_0)^{r-1} (c - c_0) - (b_0 + p c_0)^r}{p^2}, \text{ for all } r \geq 0.$$

The  $n$  first terms of  $b$  can be computed with  $O(\log p |n \log p| \log n)$  bit-operations.

**Proof.** As a shorthand we let  $\beta = b_0 + p c_0$  and  $d = c - c_0$ . Equation (II.8) simply follows from

$$\begin{aligned} b^p - a &= (\beta + p d)^p - a \\ &= p^2 \gamma_p + p^2 \beta^{p-1} d + \beta^p - a \\ &= p^2 \gamma_p + p^2 \beta^{p-1} d - p^2 \tilde{a} = 0. \end{aligned}$$

Since

$$\gamma_p = \sum_{i=2}^p \binom{p}{i} \beta^{p-i} p^{i-2} d^i,$$

and since  $d$  has positive valuation, equation (II.8) actually defines  $c$  as a recursive number.

As in the regular case, we need to provide an efficient way to compute  $\gamma_r$ . We proceed by induction on  $r$ . If  $r = 1$ , then  $\gamma_r = 0$ . If  $r = 2$ , then  $\gamma_r = d^2$ , which preserves the recursiveness. Assume now that  $r = 2h$  and that  $\gamma_h$  is available by induction. From

$$\begin{aligned} (p^2 \gamma_h)^2 &= (\beta + p d)^r + (h p \beta^{h-1} d + \beta^h)^2 \\ &\quad - 2 (h p \beta^{h-1} d + \beta^h) (p^2 \gamma_h + h p \beta^{h-1} d + \beta^h) \\ &= (\beta + p d)^r - (h p \beta^{h-1} d + \beta^h)^2 - 2 (h p \beta^{h-1} d + \beta^h) p^2 \gamma_h, \end{aligned}$$

we deduce that

$$\begin{aligned} p^2 \gamma_r &= (p^2 \gamma_h)^2 + (h p \beta^{h-1} d + \beta^h)^2 + 2 (h p \beta^{h-1} d + \beta^h) p^2 \gamma_h - r p \beta^{r-1} d - \beta^r \\ &= p^2 \gamma_h (p^2 \gamma_h + 2 (h p \beta^{h-1} d + \beta^h)) + (h p \beta^{h-1} d)^2, \end{aligned}$$

whence

$$\gamma_r = \gamma_h (p^2 \gamma_h + 2 (h p \beta^{h-1} d + \beta^h)) + (h \beta^{h-1} d)^2.$$

Since  $\gamma_h$  and  $d$  have positive valuation, the recursiveness is well preserved through this intermediate expression.

On the other hand, if  $r$  is odd, then we can write  $r = h + 1$ , with  $h$  even, and assume that  $\gamma_h$  is available by induction. Then we have that:

$$\begin{aligned} p^2 \gamma_r &= p^2 (\beta + p d) \gamma_h + (\beta + p d) (h p \beta^{h-1} d + \beta^h) - (h + 1) p \beta^h d - \beta^{h+1} \\ &= p^2 (\beta + p d) \gamma_h + h p^2 \beta^{h-1} d^2, \end{aligned}$$

whence

$$\gamma_r = (\beta + p d) \gamma_h + h \beta^{h-1} d^2,$$

which again preserves the recursiveness. Finally the equation of  $d$  can be evaluated with  $O(\log r)$  products and one division, which concludes the proof by Propositions II.7 and II.11.  $\square$

**Remark II.19.** As in the regular case, we can see  $c$  as a  $p^k$ -recursive number as soon as  $c$  is known modulo  $p^k$ . In fact, letting  $C = C_0 + D$ , with  $C_0 = C \bmod p^k$ , equation (II.8) rewrites into

$$C = C_0 + \frac{\tilde{A} - \Gamma_p}{(b_0 + p C_0)^{p-1}},$$

where  $\tilde{A} = (a - (b_0 + p C_0)^p)/p^2$ , and

$$\Gamma_r = \frac{(b_0 + p C)^r - r p (b_0 + p C_0)^{r-1} (C - C_0) - (b_0 + p C_0)^r}{p^2}, \text{ for all } r \geq 0.$$

Notice that division by  $p^2$  in base  $p^k$ , with  $k > 2$ , is equivalent to multiplication by  $p^{k-2}$  and division by  $p^k$ , which corresponds to a simple shift. Then  $\Gamma_p$  can be computed by recurrence with the same formula as  $\gamma_p$ , *mutatis mutandis*. In this way  $C$  is  $p^k$ -recursive, so that we can naturally benefit from the monoblock strategy of Section 4.2.

## 7.5 Timings

In the following table, we give the computation time of the square root using our fast relaxed product of Section 4.3 that has been reported in Table II.5. Since, in terms of performances, the situation is very similar to the division, we only compare to the zealous implementation in PARI/GP version 2.3.5. Furthermore, with MAPLE 14, the computation of a square root of a nonnegative integer lesser than  $p^8$ , *via sqrt* or *rootp* took more than 3 minutes.

$n$	8	16	32	64	128	256	512	1024	2048
blocks Binary_Mul_Padic <sub>p</sub>	14	22	39	91	230	520	1100	2400	5400
PARI/GP	5	11	28	74	210	670	2300	8100	30000

**Table II.9.** Square root, for  $p = 536870923$ , in microseconds.

## 8 Conclusion

From more than a decade a major stream in complexity theory for computer algebra has spread the idea that high level algorithms must be parameterized in terms of a small number of elementary operations (essentially integer, polynomial and matrix multiplication), so that the main goal in algorithm design consists of reducing as fast as possible to these operations. Although many asymptotically efficient algorithms have been developed along these lines, an overly doctrinaire application of this philosophy tends to be counterproductive.

For example, when it comes to computations in completions, we have seen that there are two general approaches: Newton's method and the relaxed (or lazy) approach. It is often believed that Newton's method is simply the best, because it asymptotically leads to the same complexity as integer or polynomial multiplication. However, this "reduction" does not take into account possible sparsity in the data, non asymptotic input sizes, more involved types of equations (such as partial differential equations), etc.

In this chapter, we have demonstrated that, in the area of computations with  $p$ -adic numbers, the relaxed approach can be more efficient than methods based on Newton iteration. The gains are sometimes important: in Tables II.6 and II.7, we have shown that Hensel lifting in high dimensions can become more than 10 times faster, when using the relaxed approach. At other moments, we were ourselves surprised: in Table II.8, we see that, even for the division of  $p$ -adic numbers, a naive implementation of the relaxed product yields better performances than a straightforward use of GMP, whenever  $p$  is sufficiently large.

Of course, the detailed analysis of the mutual benefits of both approaches remains an interesting subject. On the one hand, Newton iteration can be improved using blockwise techniques [Ber00, Hoe10]. On the other hand, the relaxed implementation can be improved for small sizes by ensuring a better transition between hardware and long integers, and massive inlining. At huge precisions, the recursive blockwise technique from [Hoe07b] should also become useful. Finally, “FFT-caching” could still be used in a more systematic way, and in particular for the computation of squares.

To conclude our comparison between Newton iteration and the relaxed approach, we would like to stress that, under most favourable circumstances, Newton iteration can only be hoped to be a small constant times faster than the relaxed approach, since the overhead  $O(\log n)$  of relaxed multiplication should really be read as  $(1/2)\log(n/128)$  or less. In other extreme cases, Newton iteration is several hundreds times slower, or even does not apply at all (*e.g.* for the resolution of partial differential equations).

Let us finally mention that the relaxed resolution of recursive systems of equations has been extended to more general systems of implicit equations [Hoe09]. The computation of such local solutions is the central task of the polynomial system solver called KRONECKER (see [DL08] for an introduction). We are confident that the results of [Hoe09], which were presented in the power series context, extend to more general completions, and that the relaxed model will lead to an important speed-up.

## Bibliography

- [Ber00] D. Bernstein. Removing redundancy in high precision Newton iteration. Manuscript available from <http://cr.yp.to/fastnewton.html>, 2000.
- [BHL11] J. Berthomieu, J. van der Hoeven and G. Lecerf. Relaxed algorithms for  $p$ -adic numbers. *J. Théor. Nombres Bordeaux*, 23(3):541–577, 2011.
- [BK78] R. P. Brent and H. T. Kung. Fast algorithms for manipulating formal power series. *Journal of the ACM*, 25:581–595, 1978.
- [Bre76] R. P. Brent. The complexity of multiprecision arithmetic. In R. S. Anderssen and R. P. Brent, editors, *Complexity of computational problem solving*, pages 126–165. Brisbane, 1976. University of Queensland Press.
- [CC90] D. V. Chudnovsky and G. V. Chudnovsky. Computer algebra in the service of mathematical physics and number theory. In *Computers in mathematics (Stanford, CA, 1986)*, volume 125 of *Lecture Notes in Pure and Appl. Math.*, pages 109–232. New-York, 1990. Dekker.

- [CK91] D. G. Cantor and E. Kaltofen. On fast multiplication of polynomials over arbitrary algebras. *Acta Inform.*, 28(7):693–701, 1991.
- [De 04] S. De Smedt.  $P$ -adic arithmetic. *The Mathematica Journal*, 9(2):349–357, 2004.
- [DL08] C. Durvy and G. Lecerf. A concise proof of the Kronecker polynomial system solver from scratch. *Expositiones Mathematicae*, 26(2):101–139, 2008.
- [FS56] A. Fröhlich and J. C. Shepherdson. Effective procedures in field theory. *Philos. Trans. Roy. Soc. London. Ser. A.*, 248:407–432, 1956.
- [Für07] M. Fürer. Faster Integer Multiplication. In *Proceedings of the Thirty-Ninth ACM Symposium on Theory of Computing (STOC 2007)*, pages 57–66. San Diego, California, 2007.
- [Gat84] J. von zur Gathen. Hensel and Newton methods in valuation rings. *Math. Comp.*, 42(166):637–661, 1984.
- [GG03] J. von zur Gathen and J. Gerhard. *Modern computer algebra*. Cambridge University Press, Cambridge, Second edition, 2003.
- [G+91] T. Granlund et al. GMP, the GNU multiple precision arithmetic library. 1991. Available from <http://www.swox.com/gmp>.
- [HH09] W. B. Hart and D. Harvey. FLINT 1.5.1: fast library for number theory. 2009. Available from <http://www.flintlib.org>.
- [H+02] J. van der Hoeven et al. Mathemagix. 2002. Available from <http://www.mathemagix.org>.
- [Hoe97] J. van der Hoeven. Lazy multiplication of formal power series. In W. W. Kuchlin, editor, *ISSAC '97: Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation*, pages 17–20. Maui, Hawaii, 1997.
- [Hoe99] J. van der Hoeven. Fast evaluation of holonomic functions. *Theoretical Computer Science*, 210:199–215, 1999.
- [Hoe01] J. van der Hoeven. Fast evaluation of holonomic functions near and in regular singularities. *J. Symbolic Comput.*, 31(6):717–743, 2001.
- [Hoe02] J. van der Hoeven. Relax, but don't be too lazy. *J. Symbolic Comput.*, 34(6):479–542, 2002.
- [Hoe07a] J. van der Hoeven. Efficient accelero-summation of holonomic functions. *J. Symbolic Comput.*, 42(4):389–428, 2007.
- [Hoe07b] J. van der Hoeven. New algorithms for relaxed multiplication. *J. Symbolic Comput.*, 42(8):792–802, 2007.
- [Hoe09] J. van der Hoeven. Relaxed resolution of implicit equations. Manuscript available from <http://hal.archives-ouvertes.fr/hal-00441977>, 2009.
- [Hoe10] J. van der Hoeven. Newton's method and FFT trading. *J. Symbolic Comput.*, 45(8):857–878, 2010.
- [Kat07] S. Katok.  *$P$ -adic analysis compared with real*, volume 37 of *Student Mathematical Library*. American Mathematical Society, Providence, RI, 2007.
- [Kob84] N. Koblitz.  *$P$ -adic numbers,  $p$ -adic analysis, and zeta-functions*, volume 58 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, Second edition, 1984.
- [Lan02] S. Lang. *Algebra*, volume 211 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, third edition, 2002.
- [PAR08] The PARI Group, Bordeaux. *PARI/GP, version 2.3.5*. The PARI Group, Bordeaux, 2008. Available from <http://pari.math.u-bordeaux.fr>.
- [So09] W. A. Stein et al. *Sage Mathematics Software (Version 4.2.1)*. The Sage Development Team, 2009. Available from <http://www.sagemath.org>.

- [SS71] A. Schönhage and V. Strassen. Schnelle Multiplikation grosser Zahlen. *Computing*, 7:281–292, 1971.
- [Wan84] P. S. Wang. Implementation of a  $p$ -adic package for polynomial factorization and other related operations. In *EUROSAM '84: Proceedings of the 1984 International Symposium on Symbolic and Algebraic Computation*, volume 174 of *Lecture Notes in Comput. Sci.*, pages 86–99. Springer, Berlin, 1984.

# Chapitre III

## Relaxed $p$ -adic Hensel lifting for algebraic systems

### Abstract

This chapter is an extension of Chapter II, Section 7.1. Given a ring  $R$  and  $(p)$  a proper principal ideal of  $R$ , we aim at solving a system of polynomial equations  $\mathbf{P} = (P_1, \dots, P_r)$  in  $R[Y_1, \dots, Y_r]^r$  in the  $p$ -adic completion  $R_p$  of  $R$ . For this, we assume that  $\mathbf{P}_0 = \mathbf{P} \bmod p$  has a regular modular root  $\mathbf{y}_0 \in (R/(p))^r$  and we lift this root into a root  $\mathbf{y}$  in  $R_p^r$ . It is a work in progress with R. LEBRETON [BL12].

## 1 Introduction

### 1.1 Preliminaries

Let  $R$  be an *effective commutative ring*, which means that algorithms are given for any ring operation and for equality test. Given a proper principal ideal  $(p)$  with  $p \in R$ , we write  $R_p$  for the completion of the ring  $R$  for the  $p$ -adic valuation. Any element  $a \in R_p$  can be written in a non unique way  $a = \sum_{i \in \mathbb{N}} a_i p^i$  with coefficients  $a_i \in R$ . Two classical examples are the completions  $\mathbb{K}[[X]]$  of the ring of polynomials  $\mathbb{K}[X]$  for the ideal  $(X)$  and  $\mathbb{Z}_p$  of the ring of integers  $\mathbb{Z}$  for the ideal  $(p)$ , with  $p$  a prime number.

Let  $S$  be the set of *regular* elements, that is of non zero divisors, in  $R$ , we denote  $K := S^{-1} R$  the total ring of fractions of  $R$  and  $K_p$  its  $p$ -adic completion. We still denote  $M$  a subset of  $R$  such that the restriction of the canonical projection  $\pi: R \rightarrow R/(p)$  to  $M$  is a bijection between  $M$  and  $R/(p)$ . We also assume that we have quotient and remainder by  $p$  functions

$$\begin{aligned} \text{quo}(\cdot, p): R &\longrightarrow R \\ \text{rem}(\cdot, p): R &\longrightarrow M, \end{aligned}$$

such that we have

$$a = \text{quo}(a, p) p + \text{rem}(a, p),$$

for all  $a \in R$ .

We recall that the complexity of the computation of the relaxed product of two elements of  $R_p$  known to precision  $n$  is denoted by  $R(n)$ . In particular, if  $R = \mathbb{K}[[X]]$ , then one has  $R(n) \in O(M(n) \log n)$  but this can be improved depending on  $\mathbb{K}$ , see [Hoe07b], if  $R = \mathbb{Z}$ , then one has  $R(n) \in O(l(n \log p) \log n)$ , as in [BHL11]. Whenever a complexity bound is given, if it depends on  $n$ , then this complexity is assumed to be for  $n$  approaching  $+\infty$ .

## 1.2 Models of computation

In this paper, we will use the straight-line program (s.l.p.) model of computation. We give a short presentation of this notion and refer to [BCS97] for more details. Let  $R$  be a ring and  $A$  a  $R$ -algebra.

A s.l.p. is merely an ordered sequence of operations between elements of  $A$ . An *operation* of *arity*  $r$  is a map from a subset  $\mathcal{D}$  of  $A^r$  to  $A$ . We usually work with the binary arithmetic operations  $+, -, \cdot, /: A^2 \rightarrow A$ . We also define the 0-ary operations  $r^c$  for  $r \in R$  whose constant output is  $r$ . We denote the set of all these operations  $R^c$ . Finally, we consider the unary scalar multiplications  $r \times \cdot$  for  $r \in R$  and we still denote  $R$  their set. We also consider the unary scalar divisions  $\cdot/s$  for  $s \in S$ , we still denote  $S$  their set. Let us fix a set of operations  $\Omega$ , usually  $\Omega = \{+, -, \cdot, /\} \cup R \cup S \cup R^c$ .

A s.l.p. starts with a number  $\ell$  of *inputs* indexed from  $-\ell + 1$  to 0. Then it has  $k$  *instructions*  $(\Gamma_1, \dots, \Gamma_k)$  with instructions  $\Gamma_i = (\omega_i; u_{i,1}, \dots, u_{i,r_i})$  where  $-\ell < u_{i,1}, \dots, u_{i,r_i} < i$  and  $r_i$  is the arity of the operation  $\omega_i \in \Omega$ . The s.l.p.  $\Gamma$  is executable in  $\mathcal{D}$  on  $a = (a_0, \dots, a_{-\ell-1})$  with *result sequence*  $b = (b_{-\ell+1}, \dots, b_k) \in A^{\ell+k}$ , if  $b_i = a_{\ell+1}$  whenever  $-\ell + 1 \leq i \leq 0$  and  $b_i = \omega_i(b_{u_{i,1}}, \dots, b_{u_{i,r_i}})$  whenever  $1 \leq i \leq k$ .

A s.l.p.  $\Gamma$  is *executable* over the algebra  $A$  on the input  $a \in A^\ell$  if for all operations  $\omega_i$ , its inputs belong to its definition set  $\mathcal{D}_i$ . Its *multiplicative complexity*  $L^*(\Gamma)$  is the number of operations  $\omega_i$  of  $\Gamma$  that belongs to  $\{\cdot, /\}$ .

**Example III.1.** Let  $R = \mathbb{Z}$ ,  $A = \mathbb{Z}[X, Y]$  and  $\Gamma$  be the s.l.p. with two inputs indexed  $-1, 0$  and

$$\Gamma_1 = (\cdot; -1, -1), \quad \Gamma_2 = (\cdot; 1, 0), \quad \Gamma_3 = (1^c), \quad \Gamma_4 = (-; 2, 3), \quad \Gamma_5 = (3 \times \cdot; 1).$$

First, its multiplicative complexity is  $L^*(\Gamma) = 2$ . Then,  $\Gamma$  is executable on  $(X, Y) \in A^2$ , and for this input it outputs

$X$	$Y$	$X^2$	$X^2 Y$	$1$	$X^2 Y - 1$	$3 X^2$
$-1$	$0$	$1$	$2$	$3$	$4$	$5$

**Remark III.2.** For the sake of simplicity, we will associate an arithmetic expression with its “canonical” s.l.p.

For example, the arithmetic expression  $\varphi: Y \mapsto Y^4 + 1$  can be represented by the s.l.p. with one input and instructions  $\Gamma_1 = (\cdot; 0, 0)$ ,  $\Gamma_2 = (\cdot; 1, 1)$ ,  $\Gamma_3 = (1^c)$ ,  $\Gamma_4 = (+; 2, 3)$ .

Of course the notion of s.l.p. is more precise than the arithmetic expression. So we will use this equivalence whenever the choice of the s.l.p. corresponding to an arithmetic expression does not matter.

### 1.3 Relaxed recursive $p$ -adic numbers

The relaxed model was motivated by its efficient implementation of recursive  $p$ -adic numbers. Let us recall that recursive functions and  $p$ -adic numbers of order  $k$  were introduced in Chapter II, Sections 1.1.1 and 5. However, we give another definition of an algebraic recursive  $p$ -adic polynomial of order 1, more suitable to this chapter.

**Definition III.3.** Let  $\Phi \in R_p[Y]$  and  $y$  a fixed point of  $\Phi$ , i.e.  $y = \Phi(y)$ , such that  $y_0 = y \bmod p$ . Let us denote, for all  $n \in \mathbb{N}^*$ ,  $\Phi^n = \underbrace{\Phi \circ \dots \circ \Phi}_{n \text{ times}}$  and  $\Phi^0 = \text{Id}$ . We say that  $\Phi$  allows the computation of recursive  $p$ -adic number  $y$  if for all  $n \in \mathbb{N}$ , we have

$$(y - \Phi^n(y_0)) \in p^{n+1} R_p.$$

**Proposition III.4.** Let  $\Phi \in R_p[Y]$  with a fixed point  $y$ . If  $\Phi(Y) = \sum_{k=0}^d c_k (Y - y_0)^k$  is such that  $\nu_p(c_1) > 0$ , where  $y_0 = y \bmod p$ , then  $\Phi$  allows the computation of  $y$ .

**Proof.** By induction on  $n$ . At first, we have  $(y - y_0) \in p R_p$  and

$$y - \Phi(y_0) = \Phi(y) - \Phi(y_0) = \sum_{k=1}^d c_k (y - y_0)^k.$$

Therefore,  $\nu_p(c_1 (y - y_0)) \geq 2$  and for all  $k \geq 2$ ,  $\nu_p(c_k (y - y_0)^k) \geq k \geq 2$ , so that  $(y - \Phi(y_0)) \in p^2 R_p$ .

Assume that for an  $n \in \mathbb{N}$ , we have  $(y - \Phi^n(y_0)) \in p^{n+1} R_p$ , then  $\nu_p(\Phi^n(y_0) - y_0) \geq 1$  and

$$\begin{aligned} y - \Phi^{n+1}(y_0) &= \Phi(y) - \Phi(\Phi^n(y_0)) \\ &= \sum_{k=1}^d c_k [(y - y_0)^k - (\Phi^n(y_0) - y_0)^k] \\ &= \sum_{k=1}^d c_k (y - \Phi^n(y_0)) \left[ \sum_{\ell=0}^{k-1} (y - y_0)^\ell (\Phi^n(y_0) - y_0)^{k-1-\ell} \right]. \end{aligned}$$

Therefore,  $\nu_p(c_1 (y - \Phi^n(y_0))) \geq n + 2$  and for all  $k \geq 2$  and for all  $\ell$ ,  $0 \leq \ell \leq k - 1$ , we have  $\nu_p((y - \Phi^n(y_0)) (y - y_0)^\ell (\Phi^n(y_0) - y_0)^{k-1-\ell}) \geq n + 1 + \ell + k - 1 - \ell \geq n + k \geq n + 2$ . Thus,  $(y - \Phi^{n+1}(y_0)) \in p^{n+2} R_p$ .  $\square$

A more general definition can be found in [Hoe09]. But in general, the preceding proposition justifies the recursive aspect of  $y$ . However, we refer to [Hoe11] for recursive power series defined by algebraic, differential equations or a combination thereof.

The definition of a recursive  $p$ -adic number is effective: given  $y_0 \in R/(p)$  a root of the reduced polynomial  $\bar{P} \in R/(p)[Y]$ , we can compute recursively  $y_1, y_2, \dots$ , thanks to Proposition III.4.

We have to be cautious with  $\Phi$  because, even if  $\Phi(y)_n$  does not depend on  $y_n$ ,  $y_n$  could still be involved in the computation of this coefficient. Here is an example.

**Warning III.5.** Take  $R_p = \mathbb{Z}_p$  for any prime number  $p$ . Let  $\Phi(Y) = Y^2 + p$ , and  $y$  be the only solution of  $Y = \Phi(Y)$  satisfying  $y_0 = 0$ . We can check that  $\Phi$  allows the computation of  $y$  since  $c_1 = 0$ .

At the first step, we find  $y_1 = 1$ . Then we compute  $\Phi(y)_2 = (y^2 + p)_2 = (y^2)_2$ . In the relaxed product algorithm, we compute  $y_0 y_2$  and  $r = (y_1 + y_2 p)(y_1 + y_2 p)$ . Then  $\Phi(y)_2 = 2 y_0 y_2 + r_0 = 0 y_2 + r_0$ . We face two problems.

First  $y_2$  is involved in the computation of  $\Phi(y)_2$ , although  $\Phi(y)_2$  does not depend on  $y_2$ . More importantly, the  $p$ -adic number  $r$  involves and *depends* on  $y_2$ . Since we do not know  $y_2$ , we must proceed otherwise.

Because of the issue introduced in Warning III.5, we need to force the shift inside  $\Phi$ . In other terms, we must explicit the fact that  $y_n$  is not required in the computation of  $(\Phi(y))_n$ . For this matter, we introduce two new operators

$$\begin{aligned} p^i \times \cdot : R_p &\longrightarrow R_p & \cdot / p^i : p^i R_p &\longrightarrow R_p \\ a &\longmapsto p^i a, & a &\longmapsto a / p^i. \end{aligned}$$

Their implementations are trivial, using the same C++-style pseudo code as in [BHL11]: the operation  $p^i \times \cdot$  corresponds to

```
class Left_Shift_Padic_rep_p ⊇ Padic_rep_p
  a: Padic_p
  i: ℕ
  constructor (ã: Padic_p, ã: ℕ)
    a := ã; i := ã
  method next()
    if n < 0 then return 0
    return a_{n-i}
```

and  $\cdot / p^i$  corresponds to

```
class Right_Shift_Padic_rep_p ⊇ Padic_rep_p
  a: Padic_p
  i: ℕ
  constructor (ã: Padic_p, ã: ℕ)
    a := ã; i := ã
  method next()
    return a_{n+i}
```

Let  $\Omega'$  be the set of operations  $\{+, -, \cdot, /, p^i \times \cdot, \cdot / p^i\} \cup R \cup S \cup R^c$ .

**Definition III.6.** Let  $\Gamma = (\Gamma_1, \dots, \Gamma_k)$  be a s.l.p. over the  $R$ -algebra  $R_p$  with  $\ell$  inputs and operations in  $\Omega'$ . If  $-\ell + 1 \leq i \leq k$  and  $-\ell + 1 \leq j \leq 0$ , the shift  $\text{sh}(\Gamma, i, j)$  of the  $i$ th operation of  $\Gamma$  with respect to its  $j$ th input is an element of  $\mathbb{Z} \cup \{+\infty\}$ . It is defined recursively on  $i$  such that:

– if  $i \leq 0$ , then

$$\text{sh}(\Gamma, i, j) = \begin{cases} 0, & \text{if } i = j, \\ +\infty, & \text{otherwise;} \end{cases}$$

- if  $i > 0$  and  $\Gamma_i = (\omega_i; u_{i,1}, \dots, u_{i,r_i})$ , then if
  - $\omega_i \in \{+, -, \cdot, /\}$  and  $\Gamma_i = (\omega_i; u, v)$ , then  $r_i = 2$  and
 
$$\text{sh}(\Gamma, i, j) = \min(\text{sh}(\Gamma, u, j), \text{sh}(\Gamma, v, j));$$
  - $\omega_i \in R^c$  and  $\Gamma_i = (\omega_i; )$ , then  $r_i = 0$  and
 
$$\text{sh}(\Gamma, i, j) = +\infty;$$
  - $\omega_i = p^s \times \cdot$  and  $\Gamma_i = (\omega_i; u)$ , then  $r_i = 1$  and
 
$$\text{sh}(\Gamma, i, j) = \text{sh}(\Gamma, u, j) + s;$$
  - $\omega_i = \cdot / p^s$  and  $\Gamma_i = (\omega_i; u)$ , then  $r_i = 1$  and
 
$$\text{sh}(\Gamma, i, j) = \text{sh}(\Gamma, u, j) - s.$$
  - $\omega_i \in R \cup S$  and  $\Gamma_i = (\omega_i; u)$ , then  $r_i = 1$  and
 
$$\text{sh}(\Gamma, i, j) = \text{sh}(\Gamma, u, j);$$

We abbreviate  $\text{sh}(\Gamma) := \text{sh}(\Gamma, 0, k)$  if  $\Gamma$  has one input.

This definition simply formalizes which terms of the  $j$ th input is involved in the result of the  $i$ th computation from a syntactic point of view.

**Proposition III.7.** *With the notation of Definition III.6, let  $\mathbf{y} = (y_1, \dots, y_r) \in (R_p)^r$  be such that  $\Gamma$  is executable on  $\mathbf{y}$  and  $a \in R$  be the result of the  $i$ th operation of  $\Gamma$  on the input  $\mathbf{y}$ . Then for all  $n \in \mathbb{N}$ , the computation of  $a_n$  involves at most the terms  $(y_j)_l$  of the  $j$ th input  $y_j$  for  $0 \leq l \leq r - \text{sh}(\Gamma, i, j - n)$ .*

**Example III.8.** We carry on with Warning III.5. For the natural s.l.p.  $\Gamma$  with one input associated to  $\Phi: Z \mapsto Z^2 + 1$ , we have  $\text{sh}(\Gamma) = 0$ . This formalizes the previous remark that the computation of  $\Phi(y)_n$  involves  $y_l$  for  $0 \leq l \leq n$ .

Now take the s.l.p. corresponding to

$$\begin{aligned} \Psi: pR_p &\longrightarrow R_p \\ Z &\longmapsto p^2 \left( \frac{Z}{p} \right)^2 + p \end{aligned}$$

(see Remark III.2). Then

$$\text{sh}(\Psi) = \text{sh} \left( p^2 \left( \frac{Z}{p} \right)^2 \right) = \text{sh} \left( \left( \frac{Z}{p} \right)^2 \right) + 2 = \text{sh} \left( \frac{Z}{p} \right) + 2 = \text{sh}(Z) + 1 = 1.$$

Moreover,  $\Psi$  is executable on the solution  $y$  of  $y = \Phi(y)$  because  $y \in pR_p$ . So this s.l.p.  $\Psi$  solves the problem raised in Warning III.5.

Thanks to this, we are now able to explicit which s.l.p.  $\Psi$  are suited to be put in the implementation of the class `Recursive_Padic_p`.

**Definition III.9.** Let  $y$  be a recursive  $p$ -adic and its recursive equation  $\Phi \in K[Y]$  with denominators not in  $pR$ . Let  $\Psi$  be a s.l.p. with one input and operations in  $\Omega'$ .

Then  $\Psi$  is said to be a shifted algorithm for  $\Phi$  and  $y_0$  if:

- $\text{sh}(\Psi) \geq 1$ ;
- $\Psi$  is executable on  $y$  over the  $R$ -algebra  $R_p$ ;
- $\Psi$  computes  $\Phi(Y)$  on the input  $Y$  over the  $R$ -algebra  $K[Y]$ .

**Remark III.10.** There is no uniqueness of a shifted operator. For example, if  $\Phi(Y) = Y^3 + p \in \mathbb{Z}[Y]$  and  $y_0 = 0$ , then

$$\begin{aligned} \Psi: pR_p &\longrightarrow R_p & \Psi_1: pR_p &\longrightarrow R_p \\ Z &\longmapsto p^3 \left( \frac{Z}{p} \right)^3 + p, & Z &\longmapsto p^2 \left( Z \left( \frac{Z}{p} \right)^2 \right) + p. \end{aligned}$$

are two distinct shifted algorithms for  $\Phi$  and  $y_0 = 0$ . Indeed  $\text{sh}(\Psi) = 2$ ,  $\text{sh}(\Psi_1) = 1$  and they are executable on  $y$  because  $y_0 = 0$ .

We have now dealt with the algorithmic issues of relaxed recursive  $p$ -adic numbers. Now, we have to assess the complexity.

**Proposition III.11.** Let  $\Psi$  be a shifted algorithm for the recursive  $p$ -adic  $y$  whose multiplicative complexity is  $L^*$ . Then the first  $n$  terms of the relaxed  $p$ -adic can be computed with asymptotically  $L^*R(n)$  arithmetic operations.

**Proof.** The cost of the computation of the first  $n$  terms of  $y$  is the cost of the evaluation of  $\Psi(y)$  in  $R_p$ . We recall that addition, subtraction, multiplication in  $R \times R_p$  and division in  $R_p \times S$  up to the precision  $n$  have an arithmetic complexity in  $O(n)$ . Scalars from  $R$  are decomposed in  $R_p$  in constant complexity. Finally, multiplication and division in  $R_p \times R_p$  are done in  $O(R(n))$  arithmetic operations (see [BHL11]).

Since the multiplicative complexity  $L^*$  of  $\Psi$  counts exactly the latest operations, we have shown that the number of arithmetic operations done in the computation of the first  $n$  terms of  $y$  is asymptotically equivalent to  $L^*R(n)$ .  $\square$

## 2 Simple root lifting of univariate polynomials

In [BHL11, Section 7], it is showed how to compute the  $r$ th root of a  $p$ -adic number  $a$  in a recursive relaxed way,  $r$  being relatively prime to  $p$ . In this section, we extend this result to the relaxed lifting of a simple root of any polynomial  $P \in R[Y]$ . Hensel's lemma ensures that from any modular simple root  $y_0 \in R/(p)$  of  $\bar{P} \in R/(p)[Y]$ , there exists a unique lifted root  $y \in R_p$  of  $P$  such that  $y = y_0 \pmod{p}$ .

From now on,  $P$  is a polynomial with coefficients in  $R$  and  $y \in R_p$  is the unique root of  $P$  lifted from the modular simple root  $y_0 \in R/(p)$ .

**Proposition III.12.** *The polynomial*

$$\Phi(Y) := \frac{P'(y_0)Y - P(Y)}{P'(y_0)} \in K[Y]$$

*allows the computation of  $y$ .*

**Proof.** It is clear that if  $P(y) = 0$  and  $P'(y_0) \neq 0$ , then

$$y = \frac{P'(y_0)y - P(y)}{P'(y_0)} = \Phi(y).$$

Furthermore, let us write  $P(Y) = P(y_0) + P'(y_0)(Y - y_0) + \tilde{P}(Y)(Y - y_0)^2$  with  $\tilde{P} \in R[Y]$ , then we have

$$\begin{aligned} \Phi(Y) &= \frac{P'(y_0)Y - (P(y_0) + P'(y_0)(Y - y_0) + \tilde{P}(Y)(Y - y_0)^2)}{P'(y_0)} \\ &= \frac{-P(y_0) + P'(y_0)y_0 + \tilde{P}(Y)(Y - y_0)^2}{P'(y_0)}. \end{aligned}$$

As  $P'(y_0) \neq 0 \pmod{p}$ , in fact  $\Phi(Y) \in R_p[Y]$  and since  $\Phi(Y) = \sum_{k=0}^d c_k (Y - y_0)^k$  with  $c_1 = 0$ ,  $\Phi$  allows the computation of  $y$ .  $\square$

In the following subsections, we will derive some shifted algorithms associated to the recursive equation  $\Phi$  depending on the representation of  $P$ .

## 2.1 Dense polynomials

We assume in this subsection that the polynomial  $P$  is given as the vector of its coefficients  $(c_0, c_1, \dots, c_d)$  in the monomial basis  $(1, Y, \dots, Y^d)$ . To have a shifted algorithm, we need to express  $\Phi(Y)$  with a positive shift. Remark, from Definition III.6, that the shift of

$$\Phi(Y) = \frac{1}{P'(y_0)} (-c_0 + (P'(y_0) - c_1)Y - c_2Y^2 - \dots - c_dY^d)$$

is 0. Here is a way to obtain a positive shift:

**Lemma III.13.** *For all positive integer  $k$ , the s.l.p.*

$$\Gamma: Z \mapsto p^k \left( \frac{Z - y_0}{p} \right)^k$$

*is executable on  $y$  and  $\text{sh}(\Gamma) = k - 1$ .*

**Proof.** Since  $y_0 = y \pmod{p}$ ,  $\frac{y - y_0}{p} \in R_p$  and  $\Gamma$  is executable on  $y$ . Furthermore, we have

$$\text{sh} \left( p^k \left( \frac{y - y_0}{p} \right)^k \right) = \text{sh} \left( \left( \frac{y - y_0}{p} \right)^k \right) + k = \text{sh} \left( \frac{y - y_0}{p} \right) + k = k - 1. \quad \square$$

That is why we will express  $P$  in the monomial basis  $(1, (Y - y_0), \dots, (Y - y_0)^d)$ .

**Algorithm III.1****Dense polynomial shifted algorithm computation**

**Input.** A polynomial  $P \in R[Y]$  with simple root  $y_0$  in  $R/(p)$

**Output.** A shifted algorithm  $\Psi$  associated to the operator  $\Phi$

1. Compute  $t[1] := \frac{Z - y_0}{p}$ .

2. **For**  $i$  **from** 2 to  $d$   
 $t[i] := t[i - 1] \cdot t[1]$ .

3. **Return**  $\frac{(P(y_0) - P'(y_0) y_0) + \left(\frac{1}{2} P''(y_0)\right) \times (p^2 \times t[2]) + \dots + \left(\frac{1}{d!} P^{(d)}(y_0)\right) \times (p^d \times t[d])}{P'(y_0)}$ .

**Proposition III.14.** *Given a polynomial  $P$  of degree  $d$  in dense representation and an approximate zero  $y_0$ , one may define a shifted algorithm  $\Psi$  associated to the operator  $\Phi$  thanks to Algorithm III.1. The precomputation of such an operator involves  $O(M(d) \log d)$  operations in  $R$ , while the evaluation of  $\Psi(y)$  to precision  $n$  can be done in  $O(d R(n))$  operations, if  $R = \mathbb{Z}$ , then the evaluation can be done in  $O(d \lfloor n \log p \rfloor \log n)$  bit-operations.*

**Proof.** First, let us prove that  $\Psi$  is a shifted algorithm for  $\Phi$ . Term by term, we have  $\text{sh}(P'(y_0)) = +\infty$ ,  $\text{sh}(P(y_0) - P'(y_0) y_0) = +\infty$  and for all  $k$ ,  $2 \leq k \leq d$ ,

$$\text{sh}\left(\frac{1}{k!} P^{(k)}(y_0) p^k \left(\frac{Z - y_0}{p}\right)^k\right) = k - 1 \geq 1.$$

Due to Lemma III.13, we can execute  $\Psi$  on  $y$  over the  $R$ -algebra  $R_p$ . Moreover  $\Phi(Y) = \Psi(Y)$  over the  $R$ -algebra  $K[Y]$  since  $\Psi(Y)$  is the Taylor expansion of  $P$  in a neighborhood of  $y_0$ .

The coefficients of  $P$  in the basis  $(1, (Y - y_0), \dots, (Y - y_0)^d)$  can be obtained from those of the shifted polynomial  $P(Y + y_0)$ , which is precomputed in  $O(M(d) \log d)$  arithmetic operations in  $R$ . Using Proposition III.11 and  $L^*(\Psi) = d$ , we deduce that the evaluation of  $\Psi(y)$  at precision  $n$  can be done in  $O(d R(n))$  operations.  $\square$

**Remark III.15.** Let us remark that if  $2, 3, \dots, d$  are invertible in  $R$ , then one can do this precomputation in  $O(M(d))$  operations in  $R$  by [BP94, Chapter 1, Section 2].

## 2.2 Polynomials as straight-line programs

In [BHL11, Proposition 7.1], the case of the polynomial  $P(Y) = Y^d - a$  was studied. Although the general concept of shifted algorithm was not introduced, an algorithm of multiplicative complexity  $O(L^*(P))$  was given. The shifts were only present in the implementation in MATHEMAGIX [H+02]. We clarify and generalize this approach to any polynomial  $P$  given as a s.l.p. and propose a shifted algorithm  $\Psi$  whose complexity is linear in  $L^*(P)$ .

In this subsection, we fix a polynomial  $P$  given as a s.l.p. with operations in  $\Omega := \{+, -, \cdot\} \cup R \cup S \cup R^c$  and multiplicative complexity  $L^* := L^*(P)$ , and an element  $y_0 \in R/(p)$ . We define the polynomials

$$T_P(Y) := P(y_0) + P'(y_0) (Y - y_0), \quad E_P(Y) := P(Y) - T_P(Y).$$

**Definition III.16.** We are going to define recursively a vector  $\tau \in R^2$  and a s.l.p.  $\varepsilon$  with operations in  $\Omega' := \{+, -, \cdot, /, p^i \times \cdot, \cdot/p^i\} \cup R \cup S \cup R^c$ . First, we initialize  $\varepsilon^0 := 0$ ,  $\tau^0 := (y_0, 1)$ . Then we define  $\varepsilon^i$  and  $\tau^i$  recursively on  $i$  with  $1 \leq i \leq k$  by:

- if  $\Gamma_i = (a^c; )$ , then  $\varepsilon^i := 0$ ,  $\tau^i := (a, 0)$ ;
- if  $\Gamma_i = (a \times \cdot; u)$ , then  $\varepsilon^i := a \times \varepsilon^u$ ,  $\tau^i := a \tau^u$ ;
- if  $\Gamma_i = (\pm; u, v)$ , then  $\varepsilon^i := \varepsilon^u \pm \varepsilon^v$ ,  $\tau^i := \tau^u \pm \tau^v$ ;
- if  $\Gamma_i = (\cdot; u, v)$  and we denote  $\tau^u = (a, b)$ ,  $\tau^v = (c, d)$ , then  $\tau^i = (ac, ad + bc)$  and

$$\begin{aligned} \varepsilon^i &= \varepsilon^u \cdot \varepsilon^v + p \times [((b \times \varepsilon^v + d \times \varepsilon^u)/p) \cdot (Z - y_0)] + (a \times \varepsilon^v + c \times \varepsilon^u) \quad (\text{III.1}) \\ &\quad + (bd) \times [p^2 \times ((Z - y_0)/p)^2]. \end{aligned}$$

In the latter expression, the multiplications denoted by  $\cdot$  are the ones between  $p$ -adics. Finally, we set  $\varepsilon_P := \varepsilon^k$  and  $\tau_P := \tau^k$  where  $k$  is the number of instructions in the s.l.p.  $P$ .

**Lemma III.17.** The s.l.p.  $\varepsilon_P$  is a shifted algorithm for  $E_P$  and  $y_0$ . Its multiplicative complexity is bounded by  $2L^* + 1$ . Also,  $\tau_P$  is the vector of coefficients of the polynomial  $T_P$  in the basis  $(1, (Y - y_0))$ .

**Proof.** Let us call  $P_i$  the  $i$ th result of the s.l.p.  $P$  on the input  $Y$  over  $R[Y]$ , with  $0 \leq i \leq k$ . We note  $E^i := E_{P_i}$  and  $T^i := T_{P_i}$  for all  $0 \leq i \leq k$ . Let us prove recursively that  $\varepsilon^i$  is a shifted algorithm for  $E^i$  and  $y_0$ , and that  $\tau^i$  is the vector of coefficients of  $T^i$  in the basis  $(1, (Y - y_0))$ .

For the initial step  $i = 0$ , we have  $P_0 = Y$  and we verify that  $E^0(Y) = \varepsilon^0(Y) = 0$  and  $T^0(Y) = y_0 + (Y - y_0)$ . The s.l.p.  $\varepsilon_0$  is executable on  $y$  over  $R_p$  and its shift is  $+\infty$ .

Now we prove the result recursively for  $i > 0$ . We detail the case when  $\Gamma_i = (\cdot; u, v)$ , the others cases being straightforward. Equation (III.1) corresponds to the last equation of

$$\begin{aligned} P_i &= P_u P_v \\ \Leftrightarrow E^i + T^i &= (E^u + T^u)(E^v + T^v) \\ \Leftrightarrow E^i &= E^u E^v + E^u T^v + T^u E^v + (T^u T^v - T^i) \\ \Leftrightarrow E^i &= E^u E^v + (P'_v(y_0) E^u + P'_u(y_0) E^v)(Y - y_0) + (P_v(y_0) E^u + P_u(y_0) E^v) \\ &\quad + P'_u(y_0) P'_v(y_0) (Y - y_0)^2. \end{aligned}$$

Moreover we have  $\tau^i = (P_u(y_0) P_v(y_0), P'_u(y_0) P_v(y_0) + P_u(y_0) P'_v(y_0))$ . The s.l.p.  $\varepsilon^i$  is executable on  $y$  over  $R_p$  because  $(\forall 0 \leq j < i, \text{sh}(\varepsilon_j) > 0) \Rightarrow (b \times \varepsilon^v(y) + d \times \varepsilon^u(y))/p \in R_p$ . Concerning the shifts, since  $\text{sh}(\varepsilon_u), \text{sh}(\varepsilon_v) > 0$ , we have

$$\begin{aligned} \text{sh}(\varepsilon^u \cdot \varepsilon^v) &= \min(\text{sh}(\varepsilon^u), \text{sh}(\varepsilon^v)) > 0, \\ \text{sh}(p \times [((b \times \varepsilon^v + d \times \varepsilon^u)/p) \cdot (Y - y_0)]) & \\ &= 1 + \min(\text{sh}((b \times \varepsilon^v + d \times \varepsilon^u)/p), \text{sh}(Y - y_0)) \\ &= 1 + \min(\min(\text{sh}(\varepsilon^u), \text{sh}(\varepsilon^v)) - 1, 0) > 0, \\ \text{sh}(a \times \varepsilon^v + c \times \varepsilon^u) &= \min(\text{sh}(\varepsilon^u), \text{sh}(\varepsilon^v)) > 0, \\ \text{sh}((bd) \times [p^2 \times ((Y - y_0)/p)^2]) &= 1. \end{aligned}$$

Altogether  $\text{sh}(\varepsilon^i) > 0$ . Then take  $i = r$  to conclude the proof.

Concerning multiplicative complexity, we slightly change  $\varepsilon^0$  such that it computes once and for all  $((Y - y_0)/p)^2$  before returning zero. Then for all instructions  $\cdot$  in the s.l.p.  $P$ , the s.l.p.  $\varepsilon_P$  gains two multiplications between  $p$ -adics (see operations  $\cdot$  in equation (III.1)). So  $L^*(\varepsilon_P) \leq 2L^* + 1$ .  $\square$

**Proposition III.18.** *Let  $P$  be a univariate polynomial over  $R_p$  given as a s.l.p. such that its multiplicative complexity is  $L^*$ , then the following algorithm  $\Psi$ :*

$$\Psi: Z \mapsto \frac{-P(y_0) + P'(y_0) y_0 - \varepsilon_P(Z)}{P'(y_0)}$$

*is a shifted algorithm for the associated  $\Phi$  and approximate zero  $y_0$  whose evaluation complexity is bounded by  $2L^* + 1$ .*

**Proof.** We have  $\Phi(Y) = \Psi(Y)$  over  $K[Y]$  because  $\Phi(Y) = [(-P(y_0) + P'(y_0) y_0) + E_P(Y)]/P'(y_0)$ . Because of Lemma III.17 and  $\nu_p(P'(y_0)) = 0$ , the s.l.p.  $\Psi$  is executable on  $y$  over  $R_p$  and its shift is positive. We conclude with  $L^*(\Psi) = L^*(\varepsilon_P) \leq 2L^* + 1$  as the division by  $P'(y_0)$  consists of the division of an element of  $R_p$  by an element of  $R$ .  $\square$

**Remark III.19.** If we add the square operation  $\cdot^2$  to the set of operations  $\Omega$  of  $P$ , we can gain a few multiplications in  $\varepsilon_P$ . In Definition III.16, if  $\Gamma_i = (\cdot^2; u)$  and  $\tau^u = (a, b)$  then put

$$\varepsilon^i = \varepsilon^u \cdot (\varepsilon^u + 2 \times (a + b \times (Z - y_0))) + b^2 \times [p^2 \times ((Z - y_0)/p)^2].$$

So we reduce the multiplicative complexity of  $\varepsilon_P$  and  $\Psi$  by the number of square operations in the s.l.p.  $P$ .

**Theorem III.20.** *Let  $P$  be a polynomial over  $R$  and  $y_0 \in R/(p)$  such that  $P(y_0) = 0 \pmod{p}$  and  $P'(y_0) \not\equiv 0 \pmod{p}$ . Denote  $y \in R_p$  the unique solution of  $P$  lifted from  $y_0$ . Assume that  $P$  is given as a s.l.p. whose multiplicative complexity is  $L^*$ . The computation to precision  $n$  of  $y$  in the relaxed model can be done in*

$$(2L^* + 1) R(n) = \tilde{O}(L^* n)$$

*arithmetic operations.*

**Proof.** By Propositions III.12 and III.18,  $y$  can be computed as a recursive  $p$ -adic number with the shifted algorithm  $\Psi$ . Proposition III.11 gives the announced complexity.  $\square$

### 3 Relaxed linear algebra over $p$ -adic numbers

As a generalization of the results of the previous section, we will lift a simple root of a system of  $r$  equations with  $r$  unknowns in Section 4. For this matter, one needs to solve a linear system with the Jacobian matrix in a relaxed way, as we describe in this section.

For any matrix  $A \in \mathcal{M}_{s \times r}(R_p)$ , we will denote by  $a_{ij}$  the coefficient of  $A$  lying on the  $i$ th row and the  $j$ th column. Furthermore,  $A$  can be seen as a  $p$ -adic matrix, *i.e.* a  $p$ -adic number whose coefficients are matrices over  $M$ , in this case, the matrix of order  $n$  will be denoted by  $A_n \in \mathcal{M}_{s \times r}(M)$ , so that  $A = \sum_{n=0}^{\infty} A_n p^n$ .

### 3.1 Inversion of a “scalar” matrix

We can generalize the remark of [BHL11, Section 6.1]: because of the propagation of the carries, the computation of the inverse of a regular  $r \times r$  matrix with coefficients in  $M$  is not immediate in the  $p$ -adic case.

We shall introduce eight new operators:

$$\begin{aligned} \text{add\_rem, add\_quo:} & \quad R_p \times R_p \longrightarrow R_p, \\ \text{Add\_rem, Add\_quo:} & \quad \mathcal{M}_{r \times s}(R_p) \times \mathcal{M}_{r \times s}(R_p) \longrightarrow \mathcal{M}_{r \times s}(R_p) \\ \text{mul\_rem, mul\_quo:} & \quad M \times R_p \longrightarrow R_p, \\ \text{Mul\_rem, Mul\_quo:} & \quad \mathcal{M}_r(M) \times \mathcal{M}_{r \times s}(R_p) \longrightarrow \mathcal{M}_{r \times s}(R_p). \end{aligned}$$

The functions `add_rem` and `Add_rem` compute the sum of their arguments as if each argument were series in  $R/(p)[[x]]$ . In other words, it computes the sum coefficient-wise, without taking care of the carry. The functions `add_quo` and `Add_quo` are defined by

$$\begin{aligned} b + a &= \text{add\_rem}(b, a) + p \text{add\_quo}(b, a) \\ &= \sum_{n=0}^{\infty} \text{rem}(b_n + a_n, p) p^n + \sum_{n=0}^{\infty} \text{quo}(b_n + a_n, p) p^{n+1}. \\ B + A &= \text{Add\_rem}(B, A) + p \text{Add\_quo}(B, A) \\ &= \sum_{n=0}^{\infty} \text{rem}(B_n + A_n, p) p^n + \sum_{n=0}^{\infty} \text{quo}(B_n + A_n, p) p^{n+1}. \end{aligned}$$

The fifth one, `mul_rem`, computes the  $p$ -adic integer whose coefficient of order  $n$  is  $\text{rem}(\beta a_n, p)$ , while the operator `mul_quo` computes the corresponding carries as well, so that

$$\begin{aligned} \beta a &= \text{mul\_rem}(\beta, a) + p \text{mul\_quo}(\beta, a) \\ &= \sum_{n=0}^{\infty} \text{rem}(\beta a_n, p) p^n + \sum_{n=0}^{\infty} \text{quo}(\beta a_n, p) p^{n+1}. \end{aligned}$$

The operators `Mul_rem` and `Mul_quo` are matricial counterparts of `mul_rem` and `mul_quo`, so that we have

$$B A = \text{Mul\_rem}(B, A) + p \text{Mul\_quo}(B, A).$$

They are computed recursively. On  $2 \times 2$  matrices, we define for all  $(i, j) \in \{1, 2\}^2$ ,

$$\begin{aligned} (\beta a)_{ij} &= \text{add\_rem}(\text{mul\_rem}(\beta_{i1}, a_{1j}), \text{mul\_rem}(\beta_{i2}, a_{2j})) \\ &\quad + p [\text{add\_quo}(\text{mul\_rem}(\beta_{i1}, a_{1j}), \text{mul\_rem}(\beta_{i2}, a_{2j})) \\ &\quad + \text{mul\_quo}(\beta_{i1}, a_{1j}) + \text{mul\_quo}(\beta_{i2}, a_{2j})]. \end{aligned}$$

Then recursively for matrices  $B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} \in \mathcal{M}_{2r}(M)$  and  $A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \in \mathcal{M}_{2r}(R_p)$ , we set

$$\begin{aligned} \text{Mul\_rem}(B, A)_{ij} &= \text{Add\_rem}(\text{Mul\_rem}(B_{i1}, A_{1j}), \text{Mul\_rem}(B_{i2}, A_{2j})), \\ \text{Mul\_quo}(B, A)_{ij} &= \text{Add\_quo}(\text{Mul\_rem}(B_{i1}, A_{1j}), \text{Mul\_rem}(B_{i2}, A_{2j})) \\ &\quad + \text{Mul\_quo}(B_{i1}, A_{1j}) + \text{Mul\_quo}(B_{i2}, A_{2j}). \end{aligned}$$

The operators `add_rem`, `add_quo`, `mul_rem` and `mul_quo` compute their outputs at precision  $n$  in  $O(n)$  arithmetic operations (see [BHL11]).

**Proposition III.21.** *Let  $B$  and  $A$  be two  $p$ -adic matrices such that  $B \in \mathcal{M}_r(M)$  and  $A \in \mathcal{M}_{r \times s}(R_p)$ , then `Mul_rem`( $B, A$ ), `Mul_quo`( $B, A$ ) and, therefore  $B A$ , can be computed to precision  $n$  in  $O(r^2 s n)$  operations, if  $R = \mathbb{Z}$ , it can be computed in  $O(r^2 s n \lceil \log p \rceil)$  bit-operations.*

**Proof.** To compute `Mul_rem`( $B, A$ ) and `Mul_quo`( $B, A$ ), we need to do  $O(r^2 s)$  operations among `add_rem`, `add_quo`, `mul_rem` and `mul_quo`.  $\square$

**Proposition III.22.** *Let  $A$  be a relaxed matrix over  $p$ -adic numbers of size  $r \times s$  and let  $B \in \mathcal{M}_r(M)$ . If  $B$  is invertible modulo  $p$ , with given inverse  $\Gamma = B^{-1} \bmod p$ , then the product  $C = B^{-1} A$  is recursive and  $C$  satisfies the equation*

$$C = \text{Mul\_rem}(\Gamma, A - p \text{Mul\_quo}(B, C)), \quad C_0 = \Gamma A_0 \bmod p.$$

Furthermore,  $C$  can be computed up until precision  $n$  using  $O(r^2 s n)$  operations.

**Proof.** From the definitions of `Mul_rem` and `Mul_quo`, it is quite clear that  $C$  is recursive. The computation of  $\Gamma$  can be performed in  $O(r^\omega)$  operations in  $R/(p)$ . Furthermore, the functions `Mul_rem` and `Mul_quo` both take  $O(r^2 s n)$  operations.

Therefore  $C$  can be computed up to precision  $n$  with  $O(r^2 s n)$  operations.  $\square$

### 3.2 Inversion of a matrix over $R_p$

We can now apply the division of matrices over  $p$ -adic integers, as in [Hoe02].

**Proposition III.23.** *Let  $A \in \mathcal{M}_{s \times r}(R_p)$  and  $B \in \mathcal{M}_r(R_p)$  be two relaxed matrices such that  $B_0$  is invertible of inverse  $\Gamma = B_0^{-1} \bmod p$ . Then the product  $C = B^{-1} A$  is recursive and satisfies the following equation:*

$$C = B_0^{-1} \left( A - p \times \left( \frac{B - B_0}{p} C \right) \right), \quad C_0 = \Gamma A_0 \bmod p.$$

Furthermore,  $C$  can be computed to precision  $n$  using  $O(r^2 s R(n))$  operations, if  $R = \mathbb{Z}$ , it can be computed using  $O(r^2 s \lceil n \log p \rceil \log n)$  bit-operations.

**Proof.** It is clear that the  $(n + 1)$ st term of  $C$  in the left hand member depends only on the  $(n + 1)$  first terms of  $A, B$  and the  $n$  first terms of  $C$ . Therefore  $C$  is a recursive matrix. As  $\frac{B - B_0}{p}$  and  $C$  are two matrices over  $R_p$ , the cost for multiplying them is bounded by  $O(r^2 s R(n))$ . Then it remains to apply Proposition III.22 for the last product.  $\square$

**Remark III.24.** One may notice that if  $B \in \mathcal{M}_r(R)$ , then the matrix product  $\frac{B-B_0}{p} C$  can be computed up to precision  $n$  linearly in  $O(r^2 s n)$  operations. Therefore, so can  $C$ .

## 4 Root lifting for locally regular algebraic systems

In this section, we aim at computing a  $p$ -adic root  $\mathbf{y} \in R_p^r$  of a polynomial system  $\mathbf{P} = (P_1, \dots, P_r) \in R[\mathbf{Y}]^r = R[Y_1, \dots, Y_r]^r$  in a relaxed recursive way. That is, we assume that  $\mathbf{P}$  has a modular root  $\mathbf{y}_0 = (y_{1,0}, \dots, y_{r,0}) \in (R/(p))^r$  which is regular, *i.e.* its Jacobian matrix  $d\mathbf{P}(\mathbf{y}_0)$  is invertible in  $\mathcal{M}_r(R/(p))$ , so that Hensel's lemma ensures the existence and the uniqueness of  $\mathbf{y} \in R_p^r$  such that  $\mathbf{P}(\mathbf{y}) = 0$  and  $\mathbf{y}_0 = \mathbf{y} \bmod p$ . From now on,  $\mathbf{P}$  is a polynomial system with coefficients in  $R$  and  $\mathbf{y} \in R_p^r$  is the unique root of  $\mathbf{P}$  lifted from the modular regular root  $\mathbf{y}_0 \in R/(p)$ .

First, Proposition III.12 extends as follows:

**Proposition III.25.** *The polynomial system*

$$\Phi(\mathbf{Y}) = d\mathbf{P}_{\mathbf{y}_0}^{-1}(d\mathbf{P}_{\mathbf{y}_0}(\mathbf{Y}) - \mathbf{P}(\mathbf{Y})) \in K[\mathbf{Y}]^r$$

*allows the computation of  $\mathbf{y}$ .*

**Proof.** This is a straightforward adaptation of the proof of Proposition III.12. By writing  $\mathbf{P}(\mathbf{Y}) = \mathbf{P}(\mathbf{y}_0) + d\mathbf{P}_{\mathbf{y}_0}(\mathbf{Y} - \mathbf{y}_0) + \sum_{i,j=1}^r \tilde{\mathbf{P}}_{i,j}(\mathbf{Y}) (Y_i - y_{i,0})(Y_j - y_{j,0})$ , for some  $\tilde{\mathbf{P}}_{i,j} \in K[\mathbf{Y}]$ , we have

$$\Phi(\mathbf{Y}) = d\mathbf{P}_{\mathbf{y}_0}^{-1} \left( -\mathbf{P}(\mathbf{y}_0) + d\mathbf{P}_{\mathbf{y}_0}(\mathbf{y}_0) + \sum_{i,j=1}^r \tilde{\mathbf{P}}_{i,j}(\mathbf{Y}) (Y_i - y_{i,0})(Y_j - y_{j,0}) \right).$$

Therefore,  $\Phi$  allows the computation of  $\mathbf{y}$ .  $\square$

As in the univariate case, we may have to introduce some shift in  $\Phi$ . In the following, we will present how to determine some shifted algorithms  $\Psi$  associated to  $\Phi$  for the approximate root  $\mathbf{y}_0$ .

### 4.1 Dense algebraic systems

We assume that each  $P_i$  is a polynomial of total degree  $d$ . We assume that  $\mathbf{P}$  is given with a dense representation, that is, each  $P_i$  is given as its vector of coefficients  $(\mathbf{c}_{\mathbf{k}})_{\mathbf{k} \in \mathbb{N}^r}$  in the monomial basis  $(\mathbf{Y}^{\mathbf{k}})_{\mathbf{k} \in \mathbb{N}^r}$ , where for all  $\mathbf{k} \in \mathbb{N}^r$ ,  $\mathbf{k} = (k_1, \dots, k_r)$ ,  $\mathbf{Y}^{\mathbf{k}} = Y_1^{k_1} \dots Y_r^{k_r}$ . And if we denote  $|\mathbf{k}| = k_1 + \dots + k_r$ , then  $|\mathbf{k}| \leq d$ .

As in the univariate case, the shift of  $\Phi(\mathbf{Y})$  is 0. We may adapt Lemma III.13 and Proposition III.14 from the univariate polynomial case to the multivariate polynomial case straightforwardly as follows. As a consequence of the following lemma, we will express  $P_i$  in the monomial basis  $((\mathbf{Y} - \mathbf{y}_0)^{\mathbf{k}})_{\mathbf{k} \in \mathbb{N}^r}$ .

**Lemma III.26.** *For all list  $\mathbf{k} \in \mathbb{N}^r$  with  $|\mathbf{k}| > 0$ , the s.l.p.*

$$\Gamma: \mathbf{Z} \mapsto p^{|\mathbf{k}|} \left( \frac{\mathbf{Z} - \mathbf{y}_0}{p} \right)^{\mathbf{k}} = p^{|\mathbf{k}|} \times \left[ \left( \frac{Z_1 - y_{1,0}}{p} \right)^{k_1} \dots \left( \frac{Z_r - y_{r,0}}{p} \right)^{k_r} \right]$$

is executable on  $\mathbf{y}$  and for all  $j$ ,  $1 \leq j \leq r$ ,

$$\text{sh}(\Gamma, r - j, 1) = \begin{cases} +\infty, & \text{if } k_j = 0, \\ |\mathbf{k}|, & \text{otherwise.} \end{cases}$$

### Algorithm III.2

#### Dense polynomial system shifted algorithm computation

**Input.** A polynomial system  $\mathbf{P} \in R[\mathbf{Y}]$  with regular root  $\mathbf{y}_0$  in  $(R/(p))^r$

**Output.** A shifted algorithm  $\Psi$  associated to the operator  $\Phi$

1. **For**  $i$  **from** 1 to  $r$

$$\text{Compute } t[i][1] := \frac{Z_i - y_{i,0}}{p}.$$

**For**  $j$  **from** 2 to  $d$

$$t[i][j] := t[i][j-1] \cdot t[i][1].$$

2. **For**  $i$  **from** 1 to  $r$

Compute

$$\begin{aligned} N_i := & \left( P_i(\mathbf{y}_0) - \sum_{j=1}^r \frac{\partial P_i}{\partial Y_j}(\mathbf{y}_0) \right) \\ & + \frac{1}{2} \sum_{j_1, j_2=1}^r \frac{\partial^2 P_i}{\partial Y_{j_1} \partial Y_{j_2}}(\mathbf{y}_0) \times (p^2 \times (t[i][j_1] \cdot t[i][j_2])) \\ & + \dots + \frac{1}{d!} \sum_{j_1, \dots, j_d=1}^r \frac{\partial^r P_i}{\partial Y_{j_1} \dots \partial Y_{j_d}}(\mathbf{y}_0) \times (p^d \times (t[i][j_1] \cdot \dots \cdot t[i][j_d])). \end{aligned}$$

3. **Return**  $\mathbf{d} \mathbf{P}_{\mathbf{y}_0}^{-1}(N_1, \dots, N_r)$ .

**Proposition III.27.** *Given a polynomial system  $\mathbf{P} = (P_1, \dots, P_r) \in R[\mathbf{Y}]$  in dense representation, such that each  $P_i$  has total degree at most  $d$ , and an approximate zero  $\mathbf{y}_0$ , one may define a shifted algorithm  $\Psi$  associated to the operator  $\Phi$  thanks to Algorithm III.2. The precomputation of such an operator involves  $O(\mathbf{M}((2d)^r) r^2 \log d)$  operations in  $R$ , while the evaluation of  $\Psi(\mathbf{y})$  to precision  $n$  can be done in  $O\left(\binom{r+d}{r} R(n)\right)$  operations and if  $R = \mathbb{Z}$ , in  $O\left(\binom{r+d}{r} (n \log p) \log n\right)$  bit-operations.*

**Proof.** By Kronecker substitution, the coefficients of each  $P_i$  in the basis  $((\mathbf{Y} - \mathbf{y}_0)^{\mathbf{k}})_{\mathbf{k} \in \mathbb{N}^r}$  can be obtained from the shifted of a univariate polynomial of degree at most  $\Delta = (2d+1)^r$  [BP94, Chapter 1, Section 8]. Therefore, it can be computed in  $O(\mathbf{M}(\Delta) \log \Delta) = O(\mathbf{M}((2d)^r) r \log d)$  arithmetic operations in  $R$ .

It remains to evaluate  $\Psi$  at  $\mathbf{y}$ . The only multiplications between  $p$ -adics are those of type  $t[1][j_1] \cdot \dots \cdot t[1][j_r]$ ,  $j_1 + \dots + j_r \leq d$ , this yields  $O\left(\binom{r+d}{r}\right)$  products. Then, the remaining products are between a  $p$ -adic number and a partial derivative which lies in  $R$ . At last, since  $\mathbf{d} \mathbf{P}_{\mathbf{y}_0} \in \mathcal{M}_r(R)$ , by Proposition III.23 and Remark III.24, the last computation can be done in  $O(r^2 n)$  operations.  $\square$

**Remark III.28.** Remark III.15 still applies, so that the precomputation can be done in  $O(\mathbf{M}((2d)^r) r)$  operations in  $R$ .

## 4.2 Algebraic systems as straight-line programs

We keep basically the same notations as in Section 2.2. Given an algebraic systems  $\mathbf{P}$ , we define

$$\mathbf{T}_{\mathbf{P}}(\mathbf{Y}) := \mathbf{P}(\mathbf{y}_0) + d \mathbf{P}_{\mathbf{y}_0}(\mathbf{Y} - \mathbf{y}_0), \quad \mathbf{E}_{\mathbf{P}}(\mathbf{Y}) := \mathbf{P}(\mathbf{Y}) - \mathbf{T}_{\mathbf{P}}(\mathbf{Y}).$$

We adapt Definition III.16 so that we may define  $\tau$  and  $\varepsilon$  for multivariate polynomials.

**Definition III.29.** We define recursively vectors  $\tau_j \in R^{r+1}$ , indexed from 0 to  $r$  and s.l.p.'s  $\varepsilon_j$  for  $j$ ,  $1 \leq j \leq r$ , with operations in  $\Omega' := \{+, -, \cdot, /, p^i \times \cdot, \cdot / p^i\} \cup R \cup S \cup R^c$ .

First we initialize  $\varepsilon_j^{-r+i} := 0$ ,  $\tau_j^{-r+i} := \left( y_{i,0}, 0, \dots, 0, \underbrace{1}_j, 0, \dots, 0 \right)$  for all  $i$ ,  $1 \leq i \leq r$ .

Then we define  $\varepsilon_j^i$  and  $\tau_j^i$  recursively on  $i$  with  $1 \leq i \leq k_j$  where  $k_j$  is the number of instructions in the s.l.p.  $P_j$ , by almost the same formulae as in Definition III.16.

Let us detail how we define recursively  $\varepsilon_j^i$  and  $\tau_j^i$  if  $\Gamma_i = (\cdot, u, v)$ :

Let  $\tau_j^u = (a_0, a_1, \dots, a_r)$  and  $\tau_j^v = (b_0, b_1, \dots, b_r)$ , then

$$\begin{aligned} \tau_j^i &= (a_0 b_0, a_0 b_1 + a_1 b_0, \dots, a_0 b_r + a_r b_0), \\ \varepsilon_j^i &= \varepsilon_j^u \varepsilon_j^v + p \times \left[ \left( \varepsilon_j^u / p \right) \cdot \left( \sum_{\ell=1}^r b_\ell \times (Z_\ell - y_{0,\ell}) \right) + \left( \sum_{\ell=1}^r a_\ell \times (Z_\ell - y_{0,\ell}) \right) \cdot \left( \varepsilon_j^v / p \right) \right] \\ &\quad + \sum_{1 \leq \ell_1, \ell_2 \leq r} a_{\ell_1} b_{\ell_2} \times [p^2 \times [((Z_{\ell_1} - y_{0,\ell_1})/p) \cdot ((Z_{\ell_2} - y_{0,\ell_2})/p)]]. \end{aligned} \quad (\text{III.2})$$

As before, we set  $\varepsilon_{P_j} := \varepsilon_j^{k_j}$  and  $\tau_{P_j} := \tau_j^{k_j}$ .

**Lemma III.30.** The s.l.p.  $\varepsilon_{\mathbf{P}} := (\varepsilon_{P_1}, \dots, \varepsilon_{P_r})$  is a shifted algorithm for  $\mathbf{E}_{\mathbf{P}}$  and  $\mathbf{y}_0$ . Its complexity is bounded by  $3L^* + \frac{r(r+1)}{2}$ . Also, if  $\mathbf{T}_{\mathbf{P}} := (T_{P_1}, \dots, T_{P_r})$ , then  $\tau_{P_j}$  is the vector of coefficients of the polynomial  $T_{P_j}$  in the basis  $(1, (Y_1 - y_{1,0}), \dots, (Y_r - y_{r,0}))$ .

**Proof.** From Lemma III.17, it is clear that  $\varepsilon_{\mathbf{P}}$  is a shifted algorithm for  $\mathbf{E}_{\mathbf{P}}$  and  $\mathbf{y}_0$  as is  $\tau_{P_i}$  the coefficients of  $T_{P_i}$  in the basis  $(1, (Y_1 - y_{1,0}), \dots, (Y_r - y_{r,0}))$ .

Concerning the multiplicative complexity, we perform the same change for  $\varepsilon_1^0$  as for  $\varepsilon^0$  in the proof of Lemma III.17 by computing  $((Y_i - y_{i,0})/p) \cdot ((Y_j - y_{j,0})/p)$  before returning zero, therefore we have to perform  $\frac{r(r+1)}{2}$  product of  $p$ -adics. Then,

Therefore, for all instruction  $\cdot$  in the s.l.p.  $P_j$ ,  $\varepsilon_{P_j}$  gains three multiplications between  $p$ -adics (see operations  $\cdot$  in equation (III.2)). So  $L^*(\varepsilon_{\mathbf{P}}) \leq 3L^* + \frac{r(r+1)}{2}$ .  $\square$

**Proposition III.31.** Let  $\mathbf{P}$  be a polynomial system of  $r$  polynomials in  $r$  variables over  $R_p$ , given as a s.l.p., such that its multiplicative complexity is  $L^*$ , then the following algorithm  $\Psi$ :

$$\Psi: \mathbf{Z} \longmapsto d \mathbf{P}_{\mathbf{y}_0}^{-1}((- \mathbf{P}(\mathbf{y}_0) + d \mathbf{P}_{\mathbf{y}_0}(\mathbf{y}_0)) - \varepsilon_{\mathbf{P}}(\mathbf{Z}))$$

is a shifted algorithm for the associated  $\Phi$  and approximate zero  $\mathbf{y}_0$  whose evaluation complexity is bounded by  $3L^* + \frac{r(r+1)}{2}$ .

**Proof.** We just need to prove the bound for the multiplicative complexity as the remaining part is straightforwardly analogous to Proposition III.18.

The evaluation of  $d \mathbf{P}_{\mathbf{y}_0}^{-1}(\cdot)$  consists of a product of the inverse of a matrix over  $R$  and of a vector over  $R_p$ , as in Proposition III.22 and Remark III.24, with  $s = 1$ , therefore  $L^*(\Psi) = L^*(\varepsilon_P) \leq 3L^* + \frac{r(r+1)}{2}$ .  $\square$

**Theorem III.32.** *Let  $\mathbf{P}$  be a system of  $r$  polynomials in  $r$  variables over  $R$  and  $\mathbf{y}_0 \in (R/(p))^r$  such that  $\mathbf{P}(\mathbf{y}_0) = 0 \pmod{p}$  and  $\det d \mathbf{P}(\mathbf{y}_0) \neq 0 \pmod{p}$ . Denote  $\mathbf{y} \in R_p^n$  the unique solution of  $\mathbf{P}$  lifted from  $\mathbf{y}_0$ . Assume that  $\mathbf{P}$  is given as a s.l.p. whose multiplicative complexity is  $L^*$ . The computation at precision  $n$  of  $\mathbf{y}$  in the relaxed model can be done in  $\left(3L^* + \frac{r(r+1)}{2}\right) R(n)$  arithmetic operations.*

**Proof.** By Propositions III.25 and III.31,  $\mathbf{y}$  can be computed as a  $p$ -adic vector with the shifted algorithm  $\Psi$ . By Proposition III.11, we obtain the announced complexity.  $\square$

## Bibliography

- [BCS97] P. Bürgisser, M. Clausen and M. A. Shokrollahi. *Algebraic complexity theory*, volume 315 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, Berlin, 1997.
- [BHL11] J. Berthomieu, J. van der Hoeven and G. Lecerf. Relaxed algorithms for  $p$ -adic numbers. *J. Théor. Nombres Bordeaux*, 23(3):541–577, 2011.
- [BL12] J. Berthomieu and R. Lebreton. Relaxed  $p$ -adic Hensel lifting for algebraic systems. Work in progress, 2012.
- [BP94] D. Bini and V. Y. Pan. *Polynomial and matrix computations. Vol. 1*. Progress in Theoretical Computer Science. Birkhäuser Boston Inc., Boston, MA, 1994. Fundamental algorithms.
- [H+02] J. van der Hoeven et al. Mathemagix. 2002. Available from <http://www.mathemagix.org>.
- [Hoe02] J. van der Hoeven. Relax, but don't be too lazy. *J. Symbolic Comput.*, 34(6):479–542, 2002.
- [Hoe07b] J. van der Hoeven. New algorithms for relaxed multiplication. *J. Symbolic Comput.*, 42(8):792–802, 2007.
- [Hoe09] J. van der Hoeven. Relaxed resolution of implicit equations. Manuscript available from <http://hal.archives-ouvertes.fr/hal-00441977>, 2009.
- [Hoe11] J. van der Hoeven. From implicit to recursive equations. Manuscript available from <http://hal.archives-ouvertes.fr/hal-00583125>, 2011.

# Chapitre IV

## Reduction of bivariate polynomials from convex-dense to dense

### Abstract

In this chapter we present a new algorithm for reducing the usual sparse bivariate factorization problems to the dense case. This reduction simply consists of computing an invertible monomial transformation that produces a polynomial with a dense size of the same order of magnitude as the size of the integral convex hull of the support of the input polynomial. This approach turns out to be very efficient in practice, as demonstrated with our implementation. This chapter is based on an article done with G. LECERF and accepted for publication [BL10].

### 1 Introduction

Let  $\mathbb{K}$  be a field. Throughout this chapter,  $F$  represents the bivariate polynomial in the variables  $X$  and  $Y$  over  $\mathbb{K}$  that we want to factor. At the present time, the best known complexity bounds for the squarefree and irreducible factorization problems are essentially obtained in terms of the *dense size* of  $F$ . This is relevant to many situations but, in many others, it is important to take the sparsity of  $F$  into account. In this chapter, we present a simple method to transform  $F$  in a way that is compatible to factorizations, but so that the dense size becomes of the same order of magnitude as the size of the integral convex hull of the support of  $F$ . In the next paragraphs, we give precise definitions for the sparse and dense sizes, state our main complexity result on support reduction, and then corollaries on factorizations.

#### 1.1 Sizes of polynomials

Let  $\mathcal{S}$  be a finite subset of points in  $\mathbb{Z}^2$ . The *bounding rectangle* of  $\mathcal{S}$  is the smallest rectangle of the form  $(o_X, o_Y) + [0, d_X] \times [0, d_Y]$  that contains  $\mathcal{S}$ , where  $o_X, o_Y \in \mathbb{Z}$  and  $d_X, d_Y \in \mathbb{N}$ . We define the *dense size* of  $\mathcal{S}$  as  $(d_X + 1)(d_Y + 1)$ . We write  $\text{Int } \mathcal{S}$  for the *integral convex hull* of  $\mathcal{S}$ , that is the set of integer points inside the convex hull of  $\mathcal{S}$  seen as a subset of  $\mathbb{R}^2$ , so that

$$\text{Int } \mathcal{S} = \mathbb{Z}^2 \cap \left\{ \sum_{e \in \mathcal{S}} t_e e \mid t_e \in \mathbb{R}_{\geq 0} \text{ and } \sum_{e \in \mathcal{S}} t_e = 1 \right\}.$$

The *convex size* of  $\mathcal{S}$  is defined as the cardinality  $|\text{Int } \mathcal{S}|$  of  $\text{Int } \mathcal{S}$ .

For our purposes it will be convenient to consider bivariate *Laurent polynomials*. Any such polynomial  $F \in \mathbb{K}[X, Y, X^{-1}, Y^{-1}]$  can be stored as a vector of nonzero terms, with each term composed of a coefficient and an exponent, regarded as a vector in  $\mathbb{Z}^2$ . This storage is usually called the *sparse representation* of  $F$ . For any  $(i, j) \in \mathbb{Z}^2$ , we let  $F_{i,j}$  denote the coefficient of  $X^i Y^j$  in  $F$ . The *support* of  $F$  is defined as

$$\text{Supp } F = \{(i, j) \in \mathbb{Z}^2 \mid F_{i,j} \neq 0\}.$$

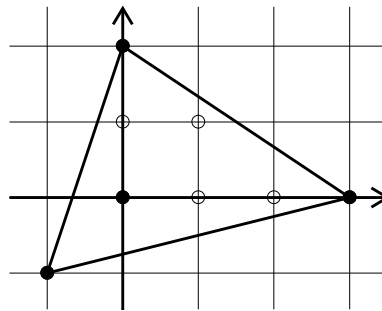
The *sparse size* of  $F$ , written as  $\sigma$ , refers to the cardinality of the support of  $F$ . We also define the *dense size* (resp. the *convex size*) of  $F$  as the dense size (resp. convex size) of its support.

The *Newton polygon* of  $F$ , written  $\text{Newton } F$ , is the convex hull of the support of  $F$  in  $\mathbb{R}^2$ . If  $F$  factors into  $GH$ , then it is known from Ostrowski [Ost21] (translated in [Ost99], and revisited later in [Ost75]) that:

$$\text{Newton } F = \text{Newton } G + \text{Newton } H = \{a + b \mid a \in \text{Newton } G, b \in \text{Newton } H\}.$$

The latter sum of the convex hulls of  $G$  and  $H$  is usually called the *Minkowski sum*. In general, even if the sparse size of  $F$  is small compared to its convex size, the irreducible factors of  $F$  can be dense with respect to their Newton polygons, what we call *convex-dense* for short. For example, simply consider  $F = Y^p - X^p \in \mathbb{Q}[X, Y]$ , where  $p$  is a prime integer: here  $\sigma = 2$  and  $F$  factors into  $X - Y$  and  $F/(X - Y)$  whose sparse size is exactly  $p$ . This shows that the irreducible factorization of  $F$  cannot be achieved in time polynomial in  $\sigma$ , and that the convex size of  $F$  is a relevant quantity to analyze the complexity of factorization problems.

**Example IV.1.** Let  $F = X^{-1}Y^{-1} + 1 + 2X^3 + 3Y^2$ . The sparse size of  $F$  is  $\sigma = 4$ . The Newton polygon of  $F$  is drawn in Figure IV.1: the black disks represent the monomials of  $F$ , while the white disks are the other monomials contained in the Newton polygon. The convex size of  $F$  is therefore  $\pi = 8$ , and since the bounding rectangle of the support of  $F$  is  $(-1, -1) + [0, 4] \times [0, 3]$ , the dense size of  $F$  is 20.



**Figure IV.1.** Newton polygon of  $F = X^{-1}Y^{-1} + 1 + 2X^3 + 3Y^2$ .

## 1.2 Main result

The method we propose in this chapter concerns all the usual types of factorization, including the squarefree, the irreducible and the absolute ones. Our main result is a pretreatment, applied to the input polynomial, which consists of a monomial trans-

formation that preserves the sparse size and roughly the convex size, but decreases the dense size. The monomial transformations considered are the maps of the affine group over  $\mathbb{Z}^2$ , written  $\text{Aff}(\mathbb{Z}^2)$ . To be precise, these are the maps  $U$

$$U: (i, j) \mapsto \begin{pmatrix} \alpha & \beta \\ \alpha' & \beta' \end{pmatrix} \begin{pmatrix} i \\ j \end{pmatrix} + \begin{pmatrix} \gamma \\ \gamma' \end{pmatrix}, \quad (\text{IV.1})$$

with  $\alpha, \beta, \gamma, \alpha', \beta'$ , and  $\gamma'$  in  $\mathbb{Z}$ , such that  $\alpha\beta' - \alpha'\beta = \pm 1$ . Such a map  $U$  preserves the absolute value of the volumes in  $\mathbb{R}^2$ .

Let  $\mathcal{S}$  be a finite subset of  $\mathbb{Z}^2$ . The set  $\mathcal{S}$  is said to be *normalized* if it belongs to  $\mathbb{N}^2$  and if it contains at least one point in  $\{0\} \times \mathbb{N}$ , and also at least one point in  $\mathbb{N} \times \{0\}$ . For such a normalized set, we write  $d_X$  for the largest abscissa involved in  $\mathcal{S}$  and, analogously,  $d_Y$  for the largest ordinate, so that the bounding rectangle is  $\mathcal{R} = [0, d_X] \times [0, d_Y]$ . The following theorem will be proven in Section 4.2:

**Theorem IV.2.** *For any normalized finite subset  $\mathcal{S}$  of  $\mathbb{Z}^2$ , of cardinality  $\sigma$ , convex size  $\pi$ , bounding rectangle  $[0, d_X] \times [0, d_Y]$ , and dense size  $\delta = (d_X + 1)(d_Y + 1)$ , one can compute an invertible affine map  $U \in \text{Aff}(\mathbb{Z}^2)$  as in (IV.1), together with  $U(\mathcal{S})$ , with  $O(\sigma \log^2 \delta)$  bit-operations, such that  $U(\mathcal{S})$  is normalized of dense size at most  $9\pi$ .*

Here, by the number of *bit-operations* we mean the size of the Boolean circuit that performs the computation, as in the *computation tree model* considered in [BCS97, Chapter 4]. The rest of this introduction is devoted to applications of Theorem IV.2 to factorizations of bivariate polynomials. Roughly speaking, given a polynomial  $F$  with a small convex size compared to its dense size, we can use the algorithm underlying Theorem IV.2 on the support of  $F$  in order to construct another polynomial to factor, with the same convex size as  $F$  but with a dense size of the same order of magnitude as the convex size. Therefore, any fast factorization algorithm in terms of the dense size leads to a fast algorithm in terms of the convex size. Another important application of our Theorem IV.2, developed by Chèze in [Chè10], concerns the decomposition of multivariate rational functions.

The proof of Theorem IV.2 is organized as follows. In our first section we explain a naive approach to reduce  $\mathcal{S}$  so that the ratio of the volumes of its convex hull and of its bounding rectangle increases. The second section provides us with a uniform bound on the latter ratio reached at the end of the reduction process. The last section is then devoted to a faster dichotomic reduction algorithm, to practical performances, and to a proof that our reduction technique leads to an essentially optimal volume ratio in the worst case.

### 1.3 Applications

We shall now explain how Theorem IV.2 can be used to reduce convex-dense factorization problems to the usual dense case. For the cost analysis we use the *computation tree model* for counting the number of operations in the ground field  $\mathbb{K}$ . Let us recall that the “soft-Oh” notation  $f(n) \in \tilde{O}(g(n))$  means that  $f(n) \in g(n) \log^{O(1)}(3 + g(n))$  (we refer the reader to [GG03, Chapter 25, Section 7] for details).

If  $U$  is an affine map of  $\mathbb{Z}^2$  as in (IV.1), then we consider its action on the monomials, and we write  $U(X^i Y^j)$  for  $X^{\alpha i + \beta j + \gamma} Y^{\alpha' i + \beta' j + \gamma'}$ . By linearity, this action extends to  $\mathbb{K}[X, Y, X^{-1}, Y^{-1}]$  as follows:

$$U(F) = \sum_{(i,j) \in \text{Supp } F} F_{i,j} U(X^i Y^j).$$

### Greatest common divisor

A Laurent polynomial is said to be *normalized* if its support is normalized. Let  $F$  and  $G$  be two normalized polynomials in  $\mathbb{K}[X, Y]$  of degree at most  $d_X$  in  $X$  and  $d_Y$  in  $Y$ , and with supports included in a common convex polygon of convex size  $\pi$ . This situation naturally occurs for instance when computing the discriminant of  $F$ , say in  $Y$ , where  $G$  is set to  $\frac{\partial}{\partial Y} F$ .

Thanks to Theorem IV.2, we can compute a reduction map  $U$  with  $O(\sigma \log^2 \delta)$  bit-operations such that the partial degrees of  $\tilde{F} = U(F)$  and  $\tilde{G} = U(G)$  are at most  $\tilde{d}_X$  in  $X$  and  $\tilde{d}_Y$  in  $Y$ , and with  $\tilde{d}_X \tilde{d}_Y \in O(\pi)$ . Without loss of generality we can further assume that  $\tilde{d}_X \geq \tilde{d}_Y$ , so that the computation of  $\tilde{H} = \text{gcd}(\tilde{F}, \tilde{G})$  in  $\mathbb{K}[X, Y]$  can be done with  $\tilde{O}(\pi^{1.5})$  operations in  $\mathbb{K}$ , assuming that  $\mathbb{K}$  has cardinality at least  $(6\tilde{d}_Y + 3)\tilde{d}_X$ , by [GG03, Corollary 11.9, part *i*]. Under the same assumptions on the cardinality of  $\mathbb{K}$ , a *randomized* variant can also obtain the same g.c.d. with an *expected number of operations* only in  $\tilde{O}(\pi)$ , by [GG03, Corollary 11.9, part *ii*].

There exists a unit  $h$  in  $\mathbb{K}[X, Y, X^{-1}, Y^{-1}]$  (that is a term  $c X^i Y^j$  with  $c$  invertible in  $\mathbb{K}$ ) such that  $H = h U^{-1}(\tilde{H})$  is normalized. We say that  $H$  is a *normalization* of  $U^{-1}(\tilde{H})$ . By the aforementioned Ostrowski theorem, it is classical to deduce that  $H$  is the actual g.c.d. of  $F$  and  $G$ , and that the convex size of  $H$  is at most  $\pi$ . Finally, the computation of  $H$  from  $\tilde{H}$  takes  $\tilde{O}(\pi \log \delta)$  more bit-operations. Of course this approach leads to a significant speedup when compared to a direct application of [GG03, Corollary 11.9] as soon as  $\pi$  is much smaller than  $\delta$ .

### Squarefree factorization

Let  $U$  be an invertible affine map over  $\mathbb{Z}^2$  as in equation (IV.1), and let  $L$  be the linear part of  $U$ . Let  $F$  still be a normalized polynomial in  $\mathbb{K}[X, Y]$  of degree at most  $d_X$  in  $X$  and  $d_Y$  in  $Y$ , of sparse size  $\sigma$ , and of convex size  $\pi$ . If the squarefree factorization of  $F$  writes into  $F = F_1^1 F_2^2 \cdots F_r^r$ , where the  $F_i$  are the pairwise coprime squarefree factors, then

$$L(F) = L(F_1)^1 L(F_2)^2 \cdots L(F_r)^r.$$

As for the g.c.d., thanks to Theorem IV.2, we can compute a reduction map  $U$  with  $O(\sigma \log^2 \delta)$  bit-operations such that the partial degrees of  $\tilde{F} = U(F)$  are at most  $\tilde{d}_X$  in  $X$  and  $\tilde{d}_Y$  in  $Y$ , and with  $\tilde{d}_X \tilde{d}_Y \in O(\pi)$ . Without loss of generality we can again assume that  $\tilde{d}_X \geq \tilde{d}_Y$ .

If  $\mathbb{K}$  has characteristic 0, then the squarefree factorization of  $\tilde{F}$  takes  $\tilde{O}(\pi^{1.5})$  operations in  $\mathbb{K}$  by [Lec08, Proposition 8]. This cost further drops to an expected one in  $\tilde{O}(\pi)$  with the randomized variant of [Lec08, Proposition 9]. Then the squarefree factors can be easily deduced by applying  $U^{-1}$  and normalizing. Other algorithms of [Lec08] concerning the separable factorization can be also adapted in the same way to the benefit of sparsity.

## Irreducible factorization

If  $F$  is a Laurent polynomial, then  $U(F)$  is irreducible if, and only if,  $F$  is irreducible. If  $F$  is normalized, then  $F$  is irreducible in  $\mathbb{K}[X, Y]$  if, and only if,  $F$  is irreducible in  $\mathbb{K}[X, Y, X^{-1}, Y^{-1}]$ . The irreducible factorization in  $\mathbb{K}[X, Y]$  can thus be deduced from the one in  $\mathbb{K}[X, Y, X^{-1}, Y^{-1}]$ . As for the squarefree factorization, we first compute a reduction map  $U$ , then we compute the irreducible factorization of  $U(F)$ , and finally we apply  $U^{-1}$  and normalize all the factors.

With this strategy, informally speaking, the algorithms of [Lec10] for instance show that the number of operations in a prime finite field can grow with only  $\tilde{O}(\pi^{1.5})$ . In Section 4.3 we report on examples that illustrate the speedup gained thanks to the reduction process.

## 1.4 Related work

Fast arithmetic operations on sparse polynomials are still a matter of active research. At the present time, the best performances are achieved essentially with supports being close to rectangles, thanks to the Kronecker substitution that reduces the product to a single variable [GG03, Chapter 8, Section 4]. Recent progress has been accomplished for instance in [HL10], but even when softly linear time algorithms are available for the sparse product, the overhead compared to dense sizes remains important. These facts motivate the strategy of the present chapter: by a direct reduction to the dense case we avoid relying on sparse arithmetic at all.

Concerning the irreducible factorization of bivariate polynomials, the Hensel lifting and recombination technique is the most popular. It leads to the best known complexity bounds [BLS+04, Lec06, Lec07, Lec10] in the dense case. Hensel lifting is used in Bernardin's implementation within MAPLE [Ber97, Ber98], and in Steel's implementation in MAGMA [Ste05, BHKS09]. In order to benefit from fast Hensel lifting, which means here with a softly linear cost, in the bivariate case, one first needs to assume that  $F$  is separable, say in  $Y$ , and then find a value  $X_0$  such that  $F(X_0, Y)$  remains separable. Unfortunately, the shift of  $X$  spoils the sparse and convex sizes. One possible solution consists of the direct computation of the irreducible factorization in  $\mathbb{K}[[X]][Y]$  but, at the present time, no algorithm with softly linear time is known for that task. Efforts have been made in this direction. For instance, in [AGL04] an algorithm for computing a factor of a given convex support is designed for special cases, with time polynomial in the convex size of the input polynomial. In [BHKS09], Puiseux series solutions of  $F$  are computed directly, with no shift in  $X$ . The best known complexity bounds for the Puiseux expansions seem to be found in [Pot08, PR11]. Recently, in [Wei10], Weimann proposed partial generalizations of the algorithms of [Lec06, Lec07]: if  $\mathbb{K}$  is a number field, and if the polynomials supported by the exterior facets of the Newton polygon are separable, then, from their irreducible factors, one can deduce the factorization with  $O(\pi^\omega)$  operations in  $\mathbb{K}$ , where  $\omega$  is the linear algebra exponent (known to be between 2 and 2.37, but unfortunately close to 3 in practice). Compared to these methods, our approach has the advantage that it can be performed from the outset with no separability assumption, that it does not need to compute the Newton polygon, and that it can benefit from fast Hensel lifting.

Another important class of factorization algorithms is due to Gao in [Gao03], who showed that the absolute factorization can be performed in softly quadratic time in terms of the dense size. The first half of his algorithm consists of computing a basis of the first De Rham cohomology group of the complementary of the hypersurface defined by  $F$ . In [GR03] it has been shown that this task can be done in time polynomial in the convex size. When a fast sparse polynomial product is available, one can even compute the probable number of absolute factors in time softly quadratic in the convex size, over finite fields with sufficiently large characteristic [HL10, Section 7]. However, these approaches still suffer an overhead when compared to the dense case, and it requires the input polynomial to be separable.

The factorization of sparse polynomials in terms of the sparse size is an active research area. Although this is not the main goal of the present chapter, let us mention briefly important results for multivariate polynomials. Polynomial time in terms of the sparse size of the output has been investigated by Zippel in [Zip79, Zip81] (see also [Zip93, Chapter 17]). Specifically, he proposed a probabilistic variant of Hensel lifting that runs in time polynomial in the total sparse size of the lifted factors of  $F$  in  $\mathbb{K}[[X]][Y]$ . His results have been extended and refined in [Gat83, Kal85, GK85, Kal89]. These techniques perform well only if the lifted factors are very sparse. Finally, another class of results focuses on the only computation of the irreducible factors of a bounded given degree. A polynomial time bound has been proved recently for this task in [AKS07] for two variables and, independently, in [KK06] directly with several variables.

## Acknowledgments

We would like to thank the anonymous referees for their helpful comments.

## 2 Support reduction

This section is devoted to the reduction algorithm underlying Theorem IV.2. We start with a naive version that is to be refined in Section 4.

### 2.1 Bounding rectangles

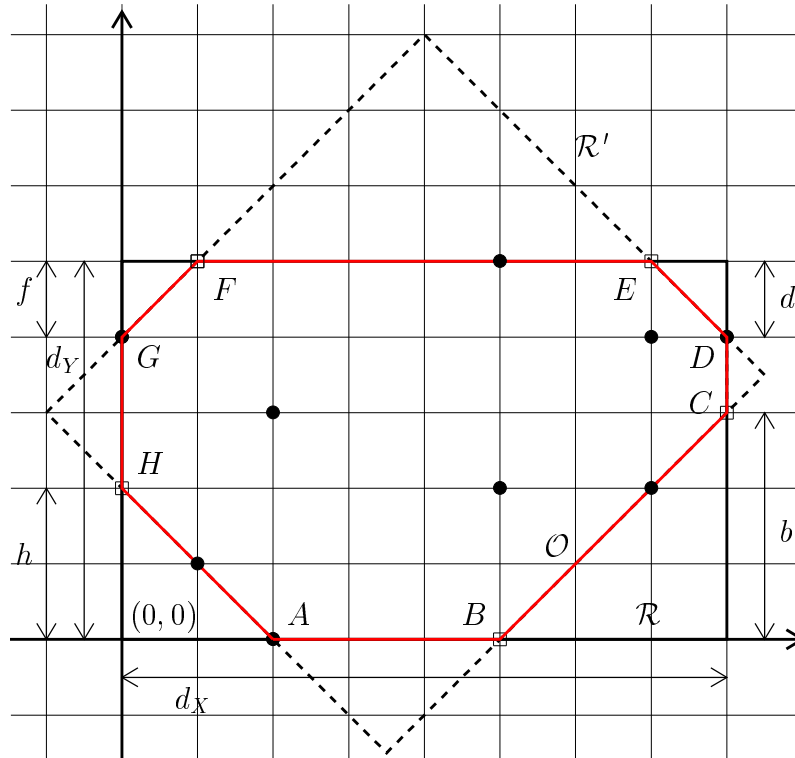
Let  $\mathcal{S}$  be a normalized finite subset of  $\mathbb{Z}^2$  with bounding rectangle  $\mathcal{R} = [0, d_X] \times [0, d_Y]$ . We introduce the integers  $b$ ,  $d$ ,  $f$  and  $h$  as follows:

- $b = d_X - \max_{(i,j) \in \mathcal{S}} (i - j)$ ,
- $d = d_X + d_Y - \max_{(i,j) \in \mathcal{S}} (i + j)$ ,
- $f = d_Y + \min_{(i,j) \in \mathcal{S}} (i - j)$ ,
- $h = \min_{(i,j) \in \mathcal{S}} (i + j)$ .

Then, let us define the following eight points, drawn in Figure IV.2 below:

$$\begin{aligned} A &= (h, 0), & B &= (d_X - b, 0), & C &= (d_X, b), & D &= (d_X, d_Y - d), \\ E &= (d_X - d, d_Y), & F &= (f, d_Y), & G &= (0, d_Y - f), & H &= (0, h). \end{aligned}$$

The rectangle  $\mathcal{R}'$  supported by lines  $(AH)$ ,  $(BC)$ ,  $(DE)$ ,  $(FG)$  is the smallest rectangle containing  $\mathcal{S}$  whose edges are parallel to the two main bisectors. The octagon  $\mathcal{O} = ABCDEFGH$  contains  $\mathcal{S}$  and any of its edges contains a point of  $\mathcal{S}$ ,  $\mathcal{O}$  is the *bounding octagon* of  $\mathcal{S}$ .



**Figure IV.2.** Bounding octagon  $\mathcal{O}$  and bounding rectangles  $\mathcal{R}$  and  $\mathcal{R}'$  for  $\mathcal{S} = \{(2, 0), (1, 1), (5, 2), (7, 2), (2, 3), (0, 4), (7, 4), (8, 4), (5, 5)\}$ .

## 2.2 Elementary transformations

Our reduction algorithm will only use the three following elementary transformations. The first one, written  $\lambda$ , corresponds to substituting  $Y/X$  into  $Y$ , this yields the following map of  $\mathbb{Z}^2$ :

$$\begin{aligned} \lambda: \mathbb{Z}^2 &\longrightarrow \mathbb{Z}^2 \\ (i, j) &\longmapsto (i - j, j). \end{aligned}$$

We will need to swap  $X$  and  $Y$ . This is the role of  $\mu$ :

$$\begin{aligned} \mu: \mathbb{Z}^2 &\longrightarrow \mathbb{Z}^2 \\ (i, j) &\longmapsto (j, i). \end{aligned}$$

Finally, translations in  $X$  are necessary to normalize the supports occurring in the reduction algorithm:

$$\begin{aligned} \tau_k: \mathbb{Z}^2 &\longrightarrow \mathbb{Z}^2 \\ (i, j) &\longmapsto (i + k, j). \end{aligned}$$

### 2.3 Reduced sets of points

Applying  $\lambda$  to  $\mathcal{S}$  modifies the volume of the bounding rectangle. For instance Figure IV.3 is the image of Figure IV.2 by  $\lambda$ : the height of  $\mathcal{R}$  does not change, but the horizontal length becomes  $d_X + d_Y - b - f$ . The points  $(i, j)$  in  $\mathcal{S}$  that are sent to the far left of  $\lambda(\mathcal{S})$  are such that  $i - j$  is minimal. Analogously, those that are sent to the far right of  $\lambda(\mathcal{S})$  are such that  $i - j$  is maximal. Applying  $\lambda^{-1}$  instead of  $\lambda$  will result in the horizontal length of the new  $\mathcal{R}$  being the difference between  $\max(i + j)$  and  $\min(i + j)$ , namely  $d_X + d_Y - d - h$ .

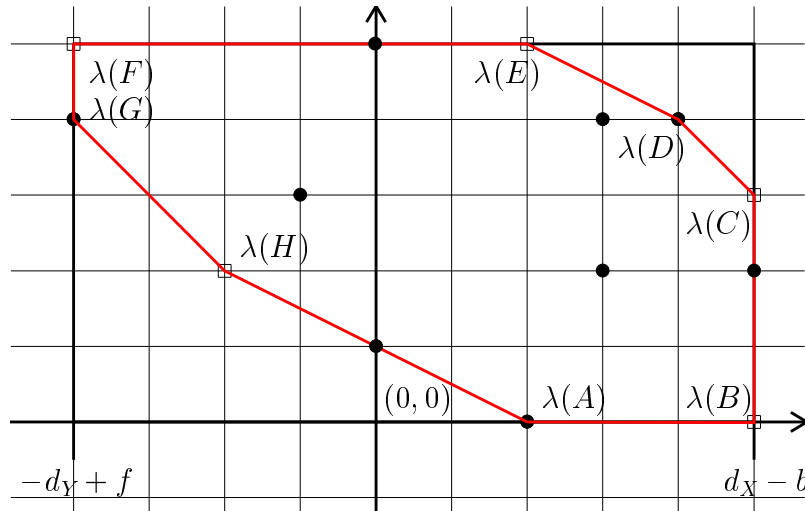


Figure IV.3. Image of the octagon of Figure IV.2 by  $\lambda$ , and its new bounding rectangle.

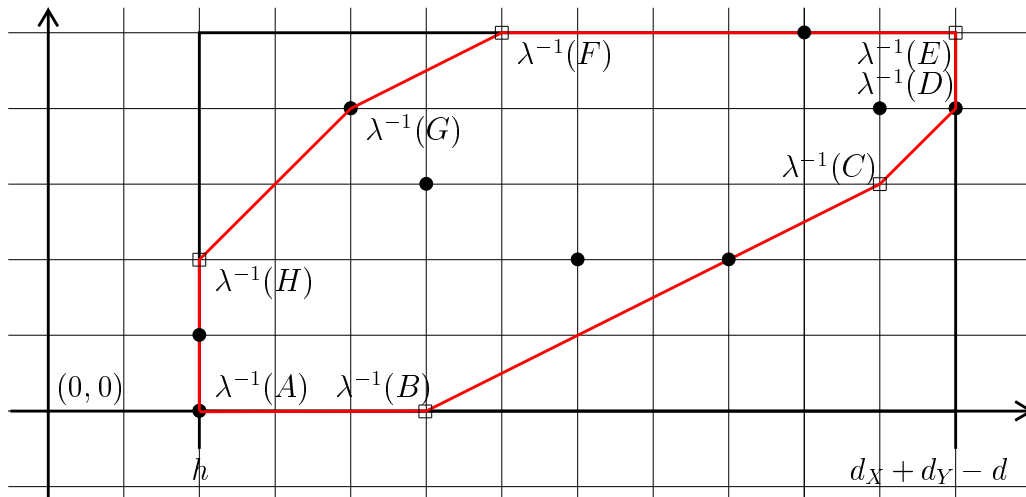


Figure IV.4. Image of the octagon of Figure IV.2 by  $\lambda^{-1}$ , and its new bounding rectangle.

From now on and until the end of this chapter,  $\eta$  represents a real number in  $[0, 3/4)$ .

**Definition IV.3.** A finite subset  $\mathcal{S}$  of  $\mathbb{Z}^2$  is said to be  $\eta$ -reduced whenever  $\mathcal{S}$  is normalized, with  $d_X$  greater than or equal to  $d_Y$ , and such that  $b, d, f$  and  $h$ , as defined in Section 2.1, satisfy the following two conditions:

1.  $b + f \leq (1 + \eta) d_Y$ , and

$$2. \quad d + h \leq (1 + \eta) d_Y.$$

If  $\mathcal{S}$  has only one point, then it is already  $\eta$ -reduced. In the next subsection, we propose an algorithm for reducing any finite subset of points of  $\mathbb{Z}^2$ . We shall see that  $\eta$  is used for controlling the tradeoff between the quality of the reduction and the time needed to reduce. The strongest reduction corresponds to  $\eta = 0$ .

## 2.4 Degenerate case

In this subsection we consider the case when  $\mathcal{S}$  is *degenerate*, which means that all the points of  $\mathcal{S}$  are aligned. If  $\mathcal{S}$  is normalized and is a singleton, then it is the origin and it is already  $\eta$ -reduced, whatever the value of  $\eta$  is. Otherwise we have the following proposition:

**Proposition IV.4.** *For any degenerate normalized finite set of points  $\mathcal{S}$  of cardinality  $\sigma$ , convex size  $\pi$ , and bounding rectangle  $[0, d_X] \times [0, d_Y]$ , one can compute an invertible affine map  $U \in \text{Aff}(\mathbb{Z}^2)$  as in (IV.1), together with  $U(\mathcal{S})$ , with  $O(\sigma \log^2 \delta)$  bit-operations, where  $\delta = (d_X + 1)(d_Y + 1)$ , such that:*

- $|\alpha|, |\beta|, |\alpha'|$ , and  $|\beta'|$  are at most  $\max(d_X, d_Y, 1)$ ,
- $|\gamma|$  and  $|\gamma'|$  are at most  $d_X d_Y$ ,
- $U(\mathcal{S})$  is normalized of dense size  $\pi$ .

**Proof.** According to the hypotheses, the following two situations can occur: the points of  $\mathcal{S}$  are either on the segment between  $(0, 0)$  and  $(d_X, d_Y)$ , or on the segment joining  $(0, d_Y)$  to  $(d_X, 0)$ . Let us first deal with the former case. Let  $g \geq 0$  be the g.c.d. of  $d_X$  and  $d_Y$ , and let  $u$  and  $v$  be the Bézout coefficients so that  $g = u d_X + v d_Y$  holds with  $|u| \leq d_Y$  and  $|v| \leq d_X$ . We refer the reader to [GG03, Lemma 3.12] for instance for these classical facts. We take  $U$  to be the linear map whose matrix is

$$\begin{pmatrix} u & v \\ -d_Y/g & d_X/g \end{pmatrix}.$$

Since  $\text{Int } \mathcal{S} = \{(i d_X/g, i d_Y/g) \mid i \in \{0, \dots, g\}\}$  we have that  $\pi = g + 1$  and that  $U(\text{Int } \mathcal{S})$  is the segment joining  $(0, 0)$  to  $(g, 0)$ . It follows that  $U(\mathcal{S})$  has dense size exactly  $\pi$ .

The latter case, where  $\mathcal{S}$  is on the segment joining  $(0, d_Y)$  to  $(d_X, 0)$ , is similar, taking

$$U: (i, j) \mapsto \begin{pmatrix} -u & v \\ d_Y/g & d_X/g \end{pmatrix} \begin{pmatrix} i \\ j \end{pmatrix} + \begin{pmatrix} u d_X \\ -d_X d_Y/g \end{pmatrix}.$$

By [GG03, Theorem 3.13] the computation of  $g$ ,  $u$ , and  $v$  can be done with  $O(\log^2 \delta)$  bit-operations using the naive version of the Euclidean algorithm. Then applying  $U$  to all the points of  $\mathcal{S}$  takes  $O(\sigma \log^2 \delta)$  bit-operations by multiplying naively the integers in quadratic time [GG03, Chapter 2, Section 3].  $\square$

We remark that the value of  $\eta$  does not intervene in this degenerate case.

## 2.5 Reduction algorithm

Until the end of this section we assume that  $\mathcal{S}$  is a nondegenerate finite set of points. The following algorithm computes  $U \in \text{Aff}(\mathbb{Z}^2)$  such that  $U(\mathcal{S})$  is  $\eta$ -reduced.

### Algorithm IV.1

#### Support reduction algorithm

**Input.** A nondegenerate normalized finite subset  $\mathcal{S}$  of  $\mathbb{N}^2$  of cardinality  $\sigma$ , and a real number  $\eta$  in  $[0, \frac{3}{4})$ .

**Output.**  $U \in \text{Aff}(\mathbb{Z}^2)$ , such that  $U(\mathcal{S})$  is  $\eta$ -reduced.

1. Compute  $(d_X, d_Y)$  for  $\mathcal{S}$ , as defined in Section 2.1.
2. Initialize  $U$  with the identity.
3. **Repeat**
  - a. **If**  $d_X < d_Y$  **then**

$$\mathcal{S} := \mu(\mathcal{S})$$

$$U := \mu \circ U$$
 Swap  $d_X$  and  $d_Y$ .
  - b. Compute  $b, d, f, h$  for  $\mathcal{S}$ , as defined in Section 2.1.
  - c. **If**  $b + f > (1 + \eta) d_Y$  **then**

$$\mathcal{S} := \tau_{d_X - f} \circ \lambda(\mathcal{S})$$

$$U := \tau_{d_X - f} \circ \lambda \circ U$$

$$d_X := d_X + d_Y - b - f$$
**else if**  $d + h > (1 + \eta) d_Y$  **then**

$$\mathcal{S} := \tau_{-h} \circ \lambda^{-1}(\mathcal{S})$$

$$U := \tau_{-h} \circ \lambda^{-1} \circ U$$

$$d_X := d_X + d_Y - d - h$$
**else return**  $U$ .

**Proposition IV.5.** *Algorithm IV.1 is correct. For any nondegenerate normalized finite subset  $\mathcal{S}$  of  $\mathbb{N}^2$  with bounding rectangle  $[0, d_X] \times [0, d_Y]$ , Algorithm IV.1 performs at most  $O(\max(d_X, d_Y))$  steps in the main **Repeat** loop.*

**Proof.** After each reduction step in the main loop, either  $d_X$  and  $d_Y$  are swapped, or  $d_X$  decreases by at least 1 and  $d_Y$  is left unchanged. Therefore the number of steps is bounded by  $O(\max(d_X, d_Y))$ . Since  $\mathcal{S}$  remains normalized all along the process, the algorithm always terminates with  $\mathcal{S}$  being  $\eta$ -reduced.  $\square$

**Example IV.6.** Assume  $\eta = 0$  and let  $F = 1 + XY + X^5 Y^2$ , whose support  $\mathcal{S}$  is  $\{(0, 0), (1, 1), (5, 2)\}$ , as drawn in Figure IV.5 below. After the first step of the algorithm, where  $\lambda$  is applied,  $\mathcal{S}$  becomes as in the left part of Figure IV.6. In the second step,  $\lambda$  is applied once more and makes  $\mathcal{S}$  reduced, as shown in the right part of Figure IV.6. In the end, the algorithm returns  $U = \tau_1 \circ \lambda^2$ , so that we have  $U(F) = X + Y + X^2 Y^2$ . The bounding rectangle of  $U(F)$  corresponds to  $d_X = d_Y = 2$ , while its bounding octagon  $\mathcal{O}$  is defined by  $b = f = h = 1$ , and  $d = 0$ .

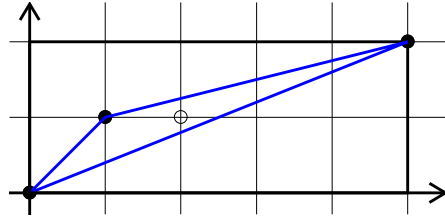


Figure IV.5. Input set  $\mathcal{S} = \{(0,0), (1,1), (5,2)\}$ .

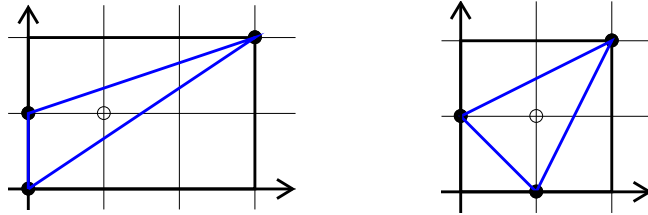


Figure IV.6.  $\mathcal{S}$  after one, and then two reduction steps.

**Example IV.7.** Let  $\mathcal{S}$  be  $\{(0,0), (1,1), (5,2)\}$  as in Example IV.6 and Figure IV.5, and assume  $\eta = 1/2$ . Since  $b + f = 4$  and  $(1 + \eta) d_Y = 3$ , the input set  $\mathcal{S}$  can be reduced by applying  $\lambda$  to obtain the same set as in the left part of Figure IV.6. However, after this reduction, we have  $b + f = 3$  which is not strictly greater than  $(1 + \eta) d_Y = 3$ . We thus see with this example that the reduction process stops earlier with  $\eta = 1/2$  than with  $\eta = 0$ .

## 2.6 Bit-cost analysis

The main difficulty in analyzing the bit-cost of Algorithm IV.1 resides in bounding the size of the entries of the map  $U$ . This is the purpose of the following lemma:

**Lemma IV.8.** *Let  $\mathcal{S}$  be a nondegenerate normalized finite subset  $\mathcal{S}$  of  $\mathbb{N}^2$  with bounding rectangle  $[0, d_X] \times [0, d_Y]$ , and let  $U$  be an affine map as in (IV.1) that sends  $\mathcal{S}$  to a normalized set  $\tilde{\mathcal{S}}$  with bounding rectangle  $[0, \tilde{d}_X] \times [0, \tilde{d}_Y]$ . Then we have:*

- $|\alpha| \leq 2 \tilde{d}_X d_Y$ ,  $|\beta| \leq 2 d_X \tilde{d}_X$ ,  $|\alpha'| \leq 2 d_Y \tilde{d}_Y$ , and  $|\beta'| \leq 2 d_X \tilde{d}_Y$ ,
- $|\gamma| \leq 4 d_X d_Y \tilde{d}_X$  and  $|\gamma'| \leq 4 d_X d_Y \tilde{d}_Y$ .

**Proof.** Since  $\mathcal{S}$  is nondegenerate, then it contains at least three points  $A = (x_A, y_A)$ ,  $B = (x_B, y_B)$ , and  $C = (x_C, y_C)$  that are not aligned. Computing the images of  $A$ ,  $B$ , and  $C$  by the linear part  $L = \begin{pmatrix} \alpha & \beta \\ \alpha' & \beta' \end{pmatrix}$  of  $U$  leads to:

$$\begin{cases} |\alpha(x_B - x_A) + \beta(y_B - y_A)| \leq \tilde{d}_X \\ |\alpha(x_C - x_A) + \beta(y_C - y_A)| \leq \tilde{d}_X. \end{cases}$$

It follows that

$$\begin{cases} |\alpha(x_B - x_A)(x_C - x_A) + \beta(y_B - y_A)(x_C - x_A)| \leq d_X \tilde{d}_X, \\ |\alpha(x_C - x_A)(x_B - x_A) + \beta(y_C - y_A)(x_B - x_A)| \leq d_X \tilde{d}_X, \end{cases}$$

whence

$$|\beta| |(y_B - y_A)(x_C - x_A) - (y_C - y_A)(x_B - x_A)| \leq 2 d_X \tilde{d}_X.$$

Since  $|(y_B - y_A)(x_C - x_A) - (y_C - y_A)(x_B - x_A)|$  is a nonzero integer, we deduce that  $|\beta| \leq 2 d_X \tilde{d}_X$ . The bounds for  $\alpha$ ,  $\alpha'$  and  $\beta'$  can be obtained *mutatis mutandis*.

Since points of the image of  $\mathcal{S}$  by  $L$  have abscissae (resp. ordinates) with absolute values at most  $4 d_X d_Y \tilde{d}_X$  (resp.  $4 d_X d_Y \tilde{d}_Y$ ), the absolute value of  $\gamma$  (resp.  $\gamma'$ ) is at most  $4 d_X d_Y \tilde{d}_X$  (resp.  $4 d_X d_Y \tilde{d}_Y$ ).  $\square$

**Proposition IV.9.** *For any nondegenerate normalized finite subset  $\mathcal{S}$  of  $\mathbb{N}^2$  of cardinality  $\sigma$ , with bounding rectangle  $[0, d_X] \times [0, d_Y]$ , and dense size  $\delta = (d_X + 1)(d_Y + 1)$ , Algorithm IV.1 takes  $O(\sigma \max(d_X, d_Y) \log \delta)$  bit-operations.*

**Proof.** Since the maximum of  $d_X$  and  $d_Y$  never increases during the main loop, the bit-size of the points in  $\mathcal{S}$  remains in  $O(\log \delta)$ . Therefore Lemma IV.8 implies that all the entries in  $U$  also remain bounded by  $O(\log \delta)$  at every step. Each reduction step thus takes  $O(\sigma \log \delta)$  bit-operations. The conclusion follows from Proposition IV.5.  $\square$

### 3 Dense size of reduced sets

Let  $\mathcal{S}$  be a finite subset of  $\mathbb{Z}^2$ . In this section, we carry on using the notation of Section 2.1, and we further write  $\text{Vol } \mathcal{S}$  for the *volume of the convex hull* of  $\mathcal{S}$ . In the next paragraphs, we show that  $\text{Vol } \mathcal{S}$  cannot be too small compared to the volume  $\text{Vol } \mathcal{R}$  of the bounding rectangle  $\mathcal{R}$  of  $\mathcal{S}$ , whenever  $\mathcal{S}$  is reduced. In the second subsection, we deduce similar bounds in terms of discrete sizes while taking care of the degenerate cases.

#### 3.1 Continuous bound

Recall that  $\eta$  is a real constant in  $[0, 3/4)$ . The following theorem guarantees that the volume spanned by an  $\eta$ -reduced set of points can be uniformly controlled in terms of the volume of its bounding rectangle:

**Theorem IV.10.** *If  $\mathcal{S}$  is an  $\eta$ -reduced set of points, then  $\text{Vol } \mathcal{S} \geq \frac{3-4\eta}{8} \text{Vol } \mathcal{R}$ , where  $\mathcal{R}$  is the bounding rectangle of  $\mathcal{S}$ .*

**Proof.** In Lemma IV.11 below, we shall show that the volume of  $\mathcal{S}$  is larger or equal to the volume of at least one of the following polygons:

$$Q_1 = ACEG,$$

$$Q_2 = BDFH,$$

$$\mathcal{P}_1 = ABDEG, \quad \mathcal{P}_2 = BCEGH,$$

$$\mathcal{P}_3 = BCEFH, \quad \mathcal{P}_4 = BDEGH,$$

$$\mathcal{P}_5 = ABDFG, \quad \mathcal{P}_6 = ACDFH,$$

$$\mathcal{P}_7 = ACEFH, \quad \mathcal{P}_8 = ACDFG.$$

Then, Lemma IV.12 asserts that  $\text{Vol } \mathcal{Q}_i \geq \frac{1-\eta}{2} \text{Vol } \mathcal{R}$ , for all  $i \in \{1, 2\}$ ; and finally, for the eight pentagons, the combination of Lemmas IV.13 and IV.15 below provides us with  $\text{Vol } \mathcal{P}_i \geq \frac{3-4\eta}{8} \text{Vol } \mathcal{R}$ , for all  $i \in \{1, \dots, 8\}$ .  $\square$

**Lemma IV.11.** *Let  $\mathcal{S}$  be a normalized finite set of points (not necessarily  $\eta$ -reduced). Then at least one of the polygons  $\mathcal{Q}_1, \mathcal{Q}_2, \mathcal{P}_1, \dots, \mathcal{P}_8$  defined above has at most  $\text{Vol } \mathcal{S}$ .*

**Proof.** From the definitions of the bounding rectangle, and of  $b, d, f, h$ , there exist eight points  $I, J, K, L, M, N, O$  and  $P$  in  $\mathcal{S}$  such that  $I \in [AB], J \in [BC], \dots, P \in [AH]$ , as drawn on the following figure (note that some of these points may coincide in particular degenerate cases):

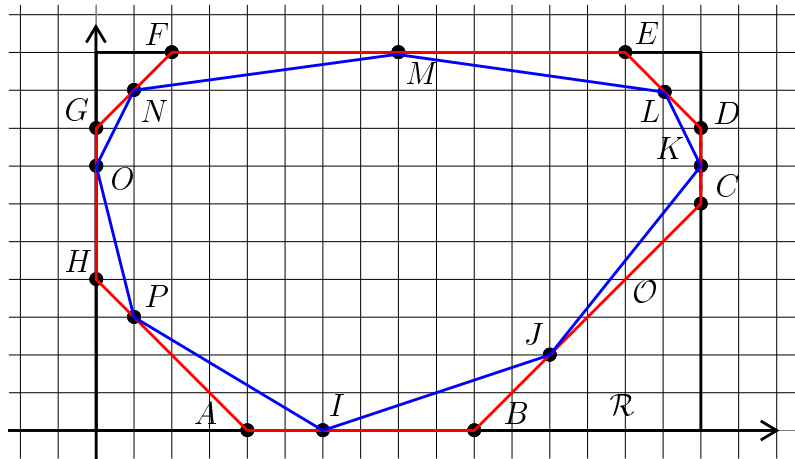


Figure IV.7. Points of  $\mathcal{S}$  lying on the bounding octagon  $\mathcal{O}$ .

Since  $\text{Vol } \mathcal{S}$  is the volume of the convex hull spanned by  $\mathcal{S}$ , it is already clear that

$$\text{Vol}(IJKLMNPO) \leq \text{Vol } \mathcal{S}.$$

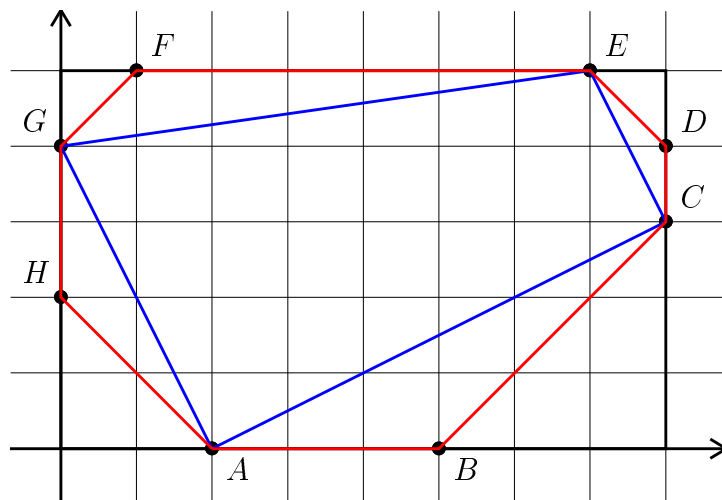
By considering the subdivision of  $IJKLMNPO$  into the triangle  $IJP$  and the polygon  $JKLMNPO$ , we see that  $\text{Vol}(AJP) \leq \text{Vol}(IJP)$  or  $\text{Vol}(BJP) \leq \text{Vol}(IJP)$ , according to the slope of  $(PJ)$  being positive or not. It follows that  $\text{Vol}(AJKLMNOP) \leq \text{Vol } \mathcal{S}$  or  $\text{Vol}(BJKLMNOP) \leq \text{Vol } \mathcal{S}$ . In other words, moving  $I$  on its supporting segment  $[A, B]$  makes  $\text{Vol}(IJKLMNPO)$  either decrease or increase. Doing so with  $K, M$  and  $O$ , and then with some points among  $J, L, N$  and  $P$ , so that  $\text{Vol}(IJKLMNPO)$  decreases, we are led to distinguish the following cases:

- If  $I, K, M$  and  $O$  all move clockwise, that is  $I$  moves to  $A, K$  moves to  $C, M$  moves to  $E$  and  $O$  moves to  $G$ , then we get the polygon  $AJCLENGP$  whose volume is at least  $\text{Vol } \mathcal{Q}_1$ .
- If  $I, K, M$  and  $O$  all move counterclockwise, then we get the polygon  $BJDLFNHP$  whose volume is at least  $\text{Vol } \mathcal{Q}_2$ .

- Otherwise two consecutive points among the cycle  $I, K, M,$  and  $O$  move into opposite directions. We now remark that the symmetries  $i \mapsto d_X - i, j \mapsto d_Y - j$  and  $(i, j) \mapsto (j, i)$  preserve the problem, the volumes, exchange the roles of  $\mathcal{Q}_1$  and  $\mathcal{Q}_2$ , and globally preserve the set of the eight pentagons  $\mathcal{P}_1, \dots, \mathcal{P}_8$ . We can thus restrict ourselves to considering for instance the case for when  $I$  moves to  $B$  and  $K$  moves to  $C$ , and examine the following subcases:
  - $M$  moves to  $E$  and  $O$  to  $H$ . If  $N$  moves to  $F$ , then we get the polygon  $BCLEFHP$ , that has volume at least  $\text{Vol } \mathcal{P}_3$ . Otherwise, if  $N$  moves to  $G$ , then we get the polygon  $BCLEGH$ , which has volume at least  $\text{Vol } \mathcal{P}_2$ .
  - $M$  moves to  $E$  and  $O$  to  $G$ . If  $P$  moves to  $H$ , then we get the polygon  $BCLENGH$ , that has volume at least  $\text{Vol } \mathcal{P}_2$ . Otherwise, if  $P$  moves to  $A$ , then we get the polygon  $BCLENGA$ , which has volume at least  $\text{Vol } \mathcal{Q}_1$ .
  - $M$  moves to  $F$  and  $O$  to  $H$ . If  $L$  moves to  $D$ , then we get the polygon  $BCDFNHP$ , that has volume at least  $\text{Vol } \mathcal{Q}_2$ . Otherwise, if  $L$  moves to  $E$ , then we get the polygon  $BCEFNHP$ , which has volume at least  $\text{Vol } \mathcal{P}_3$ .
  - $M$  moves to  $F$  and  $O$  to  $G$ . Let us assume that  $P$  moves to  $A$ . Then if  $L$  moves to  $D$ , then we get the polygon  $BCDFNGA$ , which has volume at least  $\text{Vol } \mathcal{P}_5$ . Otherwise, if  $L$  moves to  $E$ , then we get the polygon  $BCEFNGA$ , that has volume at least  $\text{Vol } \mathcal{Q}_1$ . The symmetries then handle the situation of  $P$  moving to  $H$  instead of  $A$ .  $\square$

**Lemma IV.12.** *If  $\mathcal{S}$  is an  $\eta$ -reduced set of points, then*

$$\text{Vol } \mathcal{Q}_i \geq \frac{1-\eta}{2} \text{Vol } \mathcal{R}, \text{ for } i \in \{1, 2\}.$$



**Figure IV.8.** Quadrangle  $\mathcal{Q}_1$  in octagon  $\mathcal{O}$  and rectangle  $\mathcal{R}$ .

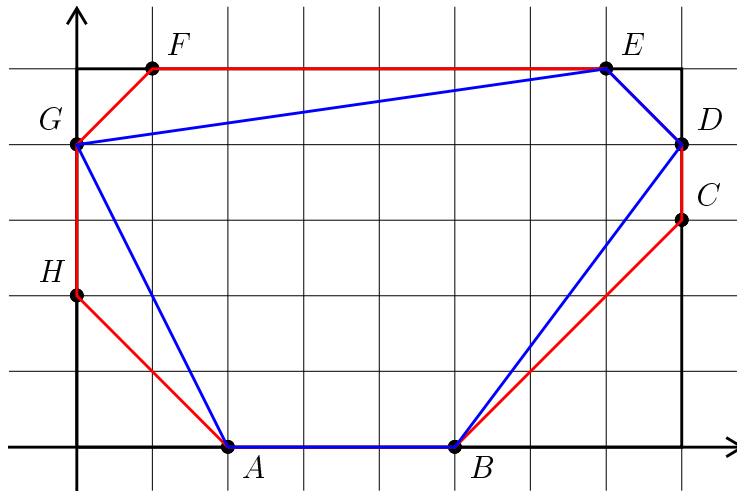
**Proof.** Since the roles of  $\mathcal{Q}_1$  and  $\mathcal{Q}_2$  are interchanged by the symmetry  $i \mapsto d_X - i$ , it suffices to prove the lemma for  $\mathcal{Q}_1$  only. We compute the volume of  $\mathcal{Q}_1$  as the difference between the volume  $d_X d_Y$  of the bounding rectangle and the volume of the four triangles outside of  $\mathcal{Q}_1$ :

$$\begin{aligned} \text{Vol } \mathcal{Q}_1 &= d_X d_Y - \frac{1}{2} ((d_X - h)b - (d_Y - b)d - (d_X - d)f - (d_Y - f)h) \\ &= \frac{1}{2} (d_Y - b - f)(d_X - d - h) + \frac{1}{2} d_X d_Y. \end{aligned} \quad (\text{IV.2})$$

Since  $\mathcal{S}$  is  $\eta$ -reduced, we have  $(1 + \eta)d_Y - b - f \geq 0$ ,  $d_X - d - h \geq 0$ , thus  $d_Y - b - f \geq -\eta d_Y$ . This yields  $\text{Vol } \mathcal{Q}_1 \geq \frac{1}{2} d_X d_Y - \frac{\eta}{2} d_X d_Y = \frac{1-\eta}{2} \text{Vol } \mathcal{R}$ .  $\square$

**Lemma IV.13.** *If  $\mathcal{S}$  is an  $\eta$ -reduced set of points, then*

$$\text{Vol } \mathcal{P}_i \geq \frac{3-4\eta}{8} \text{Vol } \mathcal{R}, \text{ for } i \in \{1, 3, 5, 7\}.$$



**Figure IV.9.** Pentagon  $\mathcal{P}_1$  in octagon  $\mathcal{O}$  and rectangle  $\mathcal{R}$ .

**Proof.** Thanks to the symmetries, it suffices to prove the lemma for  $\mathcal{P}_1$ . The volume of  $\mathcal{P}_1$  is computed as the difference of the volume of  $\mathcal{R}$  with those of the four triangles outside of  $\mathcal{P}_1$ :

$$\text{Vol } \mathcal{P}_1 = \text{Vol } \mathcal{R} - \frac{1}{2} (b(d_Y - d) + d^2 + f(d_X - d) + h(d_Y - f)).$$

From (IV.2) we deduce that:

$$\text{Vol } \mathcal{P}_1 - \text{Vol } \mathcal{Q}_1 = \frac{1}{2} (b(d_X - d_Y - h) + d(d_Y - d)).$$

Then, from

$$4b(d_X - d_Y - h) + d_Y^2 = 4b(d_X - b - h) + (2b - d_Y)^2,$$

and  $d_X - b - h \geq 0$ , it follows that  $4b(d_X - d_Y - h) + d_Y^2 \geq 0$ , and that  $\text{Vol } \mathcal{P}_1 - \text{Vol } \mathcal{Q}_1 \geq -\frac{1}{8} d_Y^2$ . The conclusion comes from Lemma IV.12:

$$\text{Vol } \mathcal{P}_1 \geq \frac{1-\eta}{2} d_X d_Y - \frac{1}{8} d_Y^2 \geq \frac{3-4\eta}{8} \text{Vol } \mathcal{R}. \quad \square$$

**Remark IV.14.** For  $\eta = 0$ , the inequality of Lemma IV.13 turns out to be sharp. For instance, with  $\mathcal{S} = \{(d_X/2, 0), (0, d_X/2), (d_X, d_X)\}$  we have  $b = f = h = \frac{1}{2} d_X$  and  $d = 0$ . Pentagon  $\mathcal{P}_1$ , as drawn on the following figure, has volume  $\frac{3}{8} d_X^2$ .

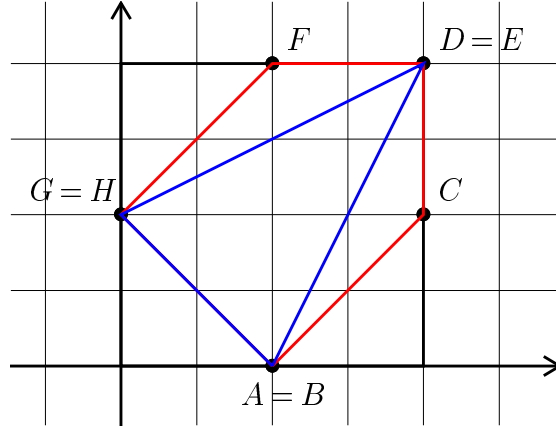


Figure IV.10. Minimal pentagon  $\mathcal{P}_1$  with  $d_X = d_Y$ .

**Lemma IV.15.** If  $\mathcal{S}$  is an  $\eta$ -reduced set of points, then

$$\text{Vol } \mathcal{P}_i \geq \frac{3-4\eta}{8} \text{Vol } \mathcal{R}, \text{ for } i \in \{2, 4, 6, 8\}.$$

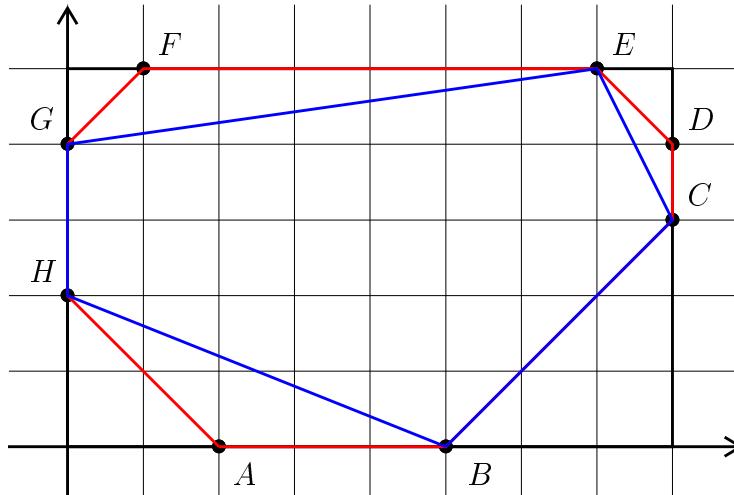


Figure IV.11. Pentagon  $\mathcal{P}_2$  in octagon  $\mathcal{O}$  and rectangle  $\mathcal{R}$ .

**Proof.** Thanks to the symmetries, it suffices to prove the lemma for  $\mathcal{P}_2$ . Specifically we shall prove that the following quantity is nonnegative:

$$\begin{aligned} \theta(b, d, f, h) &= 8 \text{Vol } \mathcal{P}_2 - (3 - 4\eta) \text{Vol } \mathcal{R} \\ &= (5 + 4\eta) d_X d_Y + 4h(b - d_X) - 4b^2 + 4d(b - d_Y) + 4f(d - d_X) \\ &= (1 + 4\eta) d_X d_Y + 4(d_X - d)(d_Y - f - h) - 4(b - d)(b - h). \end{aligned}$$

Since  $f + h \leq d_Y$ , we have that  $(1 + 4\eta) d_X d_Y + 4(d_X - d)(d_Y - f - h) \geq 0$ . Therefore, if  $h \geq b \geq d$  or  $d \geq b \geq h$ , then the lemma is proved.

Otherwise, if  $b \leq d$  and  $b \leq h$ ,  $|b - d| |b - h|$  is maximal for  $b = 0$  and, for  $d = h = (1 + \eta) d_Y / 2$ , since  $d + h \leq (1 + \eta) d_Y$ . It follows that  $-4(b - d)(b - h) \geq -(1 + \eta)^2 d_Y^2$ . From  $d_X \geq d_Y$  and  $\eta \in [0, 3/4)$  we deduce that  $(1 + 4\eta) d_X \geq (1 + \eta)^2 d_Y$ , and that  $\theta(b, d, f, h) \geq 0$ .

It remains to study the case for when  $b \geq d$  and  $b \geq h$ . Using  $d_X - d \geq b - d$ , we obtain:

$$\theta(b, d, f, h) \geq (1 + 4\eta) d_X d_Y + 4(b - d)(d_Y - b - f).$$

Then applying  $b + f \leq (1 + \eta) d_Y$  leads to:

$$\begin{aligned} \theta(b, d, f, h) &\geq (1 + 4\eta) d_X d_Y - 4(b - d) \eta d_Y \\ &\geq (1 + 4\eta) d_X d_Y - 4\eta d_X d_Y \geq 0, \end{aligned}$$

which concludes the proof.  $\square$

### 3.2 Discrete bound

For our algorithmic purposes, we need to control the number of integral points in the convex hull, instead of its volume.

**Proposition IV.16.** *If  $\mathcal{S}$  is an  $\eta$ -reduced subset of  $\mathbb{N}^2$  of convex size  $\pi$  and with bounding rectangle  $\mathcal{R} = [0, d_X] \times [0, d_Y]$ , then the following inequalities hold:*

$$\frac{3 - 4\eta}{18} (d_X + 1)(d_Y + 1) \leq \pi \leq (d_X + 1)(d_Y + 1).$$

**Proof.** As  $\mathcal{R}$  contains  $\mathcal{S}$ , the convex size  $\pi$  is always at most  $(d_X + 1)(d_Y + 1)$ . If  $\mathcal{S}$  is degenerate, then  $d_Y = 0$  and  $\pi = d_X + 1$ , so that the proposition is correct. Let us now assume that  $\mathcal{S}$  is nondegenerate. We decompose  $\text{Int } \mathcal{S}$  into  $\text{Int}_b \mathcal{S} \cup \text{Int}_i \mathcal{S}$ , where  $\text{Int}_b \mathcal{S}$  are the points lying upon the boundary of the Newton polygon of  $\mathcal{S}$ , while  $\text{Int}_i \mathcal{S}$  are the other ones strictly inside. Pick's Theorem (see [Cox69, Chapter 13, Proposition 51] or [GS93]) relates  $\text{Vol } \mathcal{S}$  to  $|\text{Int}_b \mathcal{S}|$  and  $|\text{Int}_i \mathcal{S}|$ , as follows:

$$\text{Vol } \mathcal{S} = \frac{1}{2} |\text{Int}_b \mathcal{S}| + |\text{Int}_i \mathcal{S}| - 1.$$

It follows that  $\pi \geq \text{Vol } \mathcal{S}$ , and that  $\pi \geq \frac{3 - 4\eta}{8} d_X d_Y$  by Theorem IV.10.

Whenever  $d_Y = 1$ , we have  $b + d + f + h \in \{0, 1, 2\}$ . If  $b + h = 1$  and  $d + f = 1$ , then from  $b + f \leq 1$  and  $d + h \leq 1$ , we can deduce that  $f = h$  which implies  $f = h = 0$  because  $f + h \leq 1$ . Therefore,  $b = d = 1$ , which is impossible since  $b + d \leq 1$ . Finally, we must have  $b + h = 0$  or  $d + f = 0$ , hence  $\pi \geq d_X + 1$ .

If  $d_Y \geq 2$ , the conclusion follows from  $d_X \geq 2(d_X + 1)/3$  and  $d_Y \geq 2(d_Y + 1)/3$ .  $\square$

**Remark IV.17.** In the case for when  $\eta = 0$ , if  $\alpha$  is such that the inequality  $\alpha |\text{Int } \mathcal{R}| \leq |\text{Int } \mathcal{S}|$  holds for every reduced finite subset  $\mathcal{S}$  in  $\mathbb{Z}^2$ , with  $\text{Vol } \mathcal{S} > 0$ , then necessarily we have that  $\alpha \leq 3/8$ . In fact, it suffices to consider the family  $\mathcal{S}_n = \{(n/2, 0), (0, n/2), (n, n)\}$  for  $n$  even. We have  $|\text{Int } \mathcal{S}_2| = 4$  and  $|\text{Int } \mathcal{S}_{n+2}| = |\text{Int } \mathcal{S}_n| + 3 \left(\frac{n}{2} + 1\right)$ , and deduce that

$$\frac{|\text{Int } \mathcal{S}_n|}{|\text{Int } \mathcal{R}_n|} = \frac{3n(n+2) + 8}{8(n+1)^2}$$

is decreasing and converges to  $3/8$ . In general, the constant  $\frac{3-4\eta}{18}$  thus may be rather pessimistic for large  $\mathcal{S}$ .

## 4 Faster reduction algorithm

The last ingredient now missing to prove Theorem IV.2 is a reduction algorithm with a number of reduction steps that grows only with the logarithm of the dense size. This is the goal of this section.

### 4.1 Dichotomic approach

This section is dedicated to a fast variant of Algorithm IV.1. We do not compute exactly the same output, however. Roughly speaking, the main idea is to determine quickly how many times  $\lambda$  or  $\lambda^{-1}$  can be applied before two consecutive swaps.

Let  $\mathcal{S}$  be a normalized finite subset of  $\mathbb{N}^2$  with bounding rectangle  $[0, d_X] \times [0, d_Y]$ , and let  $q$  be a positive integer. The points  $(i, j)$  in  $\mathcal{S}$  that are sent to the far left of  $\lambda^q(\mathcal{S})$  are such that  $i - qj$  is minimal. Analogously, those that are sent to the far right of  $\lambda^q(\mathcal{S})$  are such that  $i - qj$  is maximal. This motivates the introduction of  $b_q, d_q, f_q$ , and  $h_q$  as

- $b_q = d_X - \max_{(i,j) \in \mathcal{S}} (i - qj)$ ,
- $d_q = d_X + q d_Y - \max_{(i,j) \in \mathcal{S}} (i + qj)$ ,
- $f_q = q d_Y + \min_{(i,j) \in \mathcal{S}} (i - qj)$ ,
- $h_q = \min_{(i,j) \in \mathcal{S}} (i + qj)$ .

For  $q = 1$ , these definitions coincide with those of  $b, d, f$ , and  $h$  of Section 2.1. The following inequalities hold:

$$\begin{aligned} b_q + d_q &\leq q d_Y, & b_q + h_q &\leq d_X, \\ f_q + h_q &\leq q d_Y, & d_q + f_q &\leq d_X. \end{aligned}$$

The height of the bounding rectangle of  $\lambda^q(\mathcal{S})$  is still  $d_Y$ , while the horizontal length becomes  $d_X + q d_Y - b_q - f_q$ . In the same manner, the horizontal length of the bounding rectangle of  $\lambda^{-q}(\mathcal{S})$  becomes  $d_X + q d_Y - d_q - h_q$ .

From now on, the reduction factor  $\eta$  is supposed to be positive, that is in  $(0, 3/4)$ . We write  $\lfloor a \rfloor$  for the integer part of  $a$  ( $\lfloor a \rfloor \leq a < \lfloor a \rfloor + 1$ ), and  $\log_2 a$  for the logarithm of  $a$  in base 2. The fast algorithm we propose summarizes as follows:

**Algorithm IV.2****Dichotomic support reduction algorithm**

**Input.** A nondegenerate normalized finite subset  $\mathcal{S}$  of  $\mathbb{N}^2$  of cardinality  $\sigma$ , and a real number  $\eta$  in  $(0, \frac{3}{4})$ .

**Output.**  $U \in \text{Aff}(\mathbb{Z}^2)$ , such that  $U(\mathcal{S})$  is  $\eta$ -reduced.

1. Compute  $(d_X, d_Y)$  for  $\mathcal{S}$ , as defined in Section 2.1.
2. Initialize  $U$  with the identity.
3. Initialize  $m$  with  $\lfloor \log_2(d_X/(\eta d_Y)) \rfloor$ .
4. **Repeat**
  - a. **If**  $d_X < d_Y$  **then**

$$\begin{aligned} \mathcal{S} &:= \mu(\mathcal{S}) \\ U &:= \mu \circ U \\ &\text{Swap } d_X \text{ and } d_Y \\ m &:= \lfloor \log_2(d_X/(\eta d_Y)) \rfloor. \end{aligned}$$
  - b. **If**  $m < 0$  **then return**  $U$ .
  - c. Compute  $b_{2^m}, d_{2^m}, f_{2^m}, h_{2^m}$  for  $\mathcal{S}$  as defined above.
  - d. **If**  $b_{2^m} + f_{2^m} > 2^m(1 + \eta)d_Y$  **then**

$$\begin{aligned} \mathcal{S} &:= \tau_{2^m d_Y - f_{2^m}} \circ \lambda^{2^m}(\mathcal{S}) \\ U &:= \tau_{2^m d_Y - f_{2^m}} \circ \lambda^{2^m} \circ U \\ d_X &:= d_X + 2^m d_Y - b_{2^m} - f_{2^m} \end{aligned}$$
**else if**  $d_{2^m} + h_{2^m} > 2^m(1 + \eta)d_Y$  **then**

$$\begin{aligned} \mathcal{S} &:= \tau_{-h_{2^m}} \circ \lambda^{-2^m}(\mathcal{S}) \\ U &:= \tau_{-h_{2^m}} \circ \lambda^{-2^m} \circ U \\ d_X &:= d_X + 2^m d_Y - d_{2^m} - h_{2^m}. \end{aligned}$$
  - e.  $m := m - 1$ .

**Proposition IV.18.** *Assume that  $\eta > 0$ . For any nondegenerate normalized finite subset  $\mathcal{S}$  of  $\mathbb{N}^2$ , of cardinality  $\sigma$  and dense size  $\delta$ , Algorithm IV.2 is correct and runs in  $O(\sigma \log^2 \delta)$  bit-operations.*

**Proof.** Let us consider that the bounding rectangle of  $\mathcal{S}$  is  $[0, d_X] \times [0, d_Y]$  at input. Without loss of generality, we can assume that  $d_X \geq d_Y$  holds in order to simplify the proof. Then we let  $\ell_0 = d_X$  and  $\ell_1 = d_Y$ , and define the sequence  $(\mathcal{S}_i)_i$  with  $\mathcal{S}_0 = \mathcal{S}$  and  $\mathcal{S}_i$  is the current value of the set just after the  $i$ th swap, that is at the end of step (a). We write  $r$  for the total number of swaps performed during the execution of the algorithm, we let  $\ell_i$  be the largest abscissa in  $\mathcal{S}_i$ , and  $m_i$  be  $\lfloor \log_2(\ell_i/(\eta \ell_{i+1})) \rfloor$ . By convention,  $\ell_{r+1}$  is the largest ordinate in  $\mathcal{S}_r$ .

For when  $i + 2 \leq r$  holds, we have that  $\ell_{i+2} \leq \ell_i - \eta \ell_{i+1}$ . By descending induction, starting with  $\ell_r \geq 1$  and  $\ell_{r-1} \geq 1$ , we shall prove that  $\ell_i \geq \varphi^{r-i-1}$ , where  $\varphi$  is the positive root  $\frac{\eta + \sqrt{4 + \eta^2}}{2} > 1$  of the characteristic equation  $x^2 - \eta x - 1 = 0$ . Since this is true for  $i = r$  and  $i = r - 1$ , and since  $\ell_i \geq \eta \ell_{i+1} + \ell_{i+2} \geq \eta \varphi^{r-i-2} + \varphi^{r-i-3} = \varphi^{r-i-1}$ , we deduce that  $d_X = \ell_0 \geq \varphi^{r-1}$ . The number of swaps  $r$  thus drops to  $O(\log d_X)$ .

By Lemma IV.8, since all the  $\mathcal{S}_i$  are normalized, each reduction step amounts to  $O(\sigma \log \delta)$  bit-operations. On the other hand, the total number of steps is  $\sum_{i=0}^r m_i$ ,

$$\sum_{i=0}^r m_i \in O\left(\sum_{i=0}^r \log_2\left(\frac{\ell_i}{\ell_{i+1}}\right) + r \log_2 \frac{1}{\eta}\right) \in O(\log \delta),$$

which concludes the cost analysis.

We shall prove that when the algorithm stops, the final value of  $\mathcal{S}$  is  $\eta$ -reduced. We now focus on what happens just after the last swap. In short, we let  $M$  be  $m_r$  and  $\mathcal{T}_{M+1}$  be  $\mathcal{S}_r$ . We denote by  $\mathcal{T}_m$  the current value of  $\mathcal{S}$  just before entering step (e), where  $m$  being the corresponding current value of  $m$ . Therefore  $\mathcal{T}_0$  corresponds to the output of the algorithm and we want to prove that it is  $\eta$ -reduced. If  $\mathcal{T}_0 = \mathcal{T}_1$ , then we are done.

If  $\mathcal{A}$  is a subset of points, then we write  $\ell_x(\mathcal{A})$  for the horizontal length of the bounding rectangle of  $\mathcal{A}$ . Let us now assume that  $\mathcal{T}_0$  is the normalization of  $\lambda(\mathcal{T}_1)$ . In this case, of course,  $\lambda^{-1}$  does not reduce  $\mathcal{T}_0$ . Let us prove that  $\lambda$  does not reduce  $\mathcal{T}_0$  either. If  $\mathcal{T}_m$  were the normalization of  $\lambda^{2^m}(\mathcal{T}_{m+1})$  for all  $m$  in  $\{0, \dots, M\}$ , then we would deduce that

$$\begin{aligned} \ell_x(\mathcal{T}_0) &\leq \ell_x(\mathcal{T}_{M+1}) - \sum_{m=0}^M 2^m \eta \ell_{r+1} = \ell_r - (2^{M+1} - 1) \eta \ell_{r+1} \\ &< \ell_r - (\ell_r / (\eta \ell_{r+1}) - 1) \eta \ell_{r+1} = \eta \ell_{r+1}, \end{aligned}$$

which is impossible. Therefore there exists a largest integer  $\mu \in \{0, \dots, M\}$  such that for all  $m$  in  $\{0, \dots, \mu - 1\}$ ,  $\mathcal{T}_m$  is the normalization of  $\lambda^{2^m}(\mathcal{T}_{m+1})$ . This yields that  $\mathcal{T}_0$  is the normalization of  $\lambda^{2^\mu - 1}(\mathcal{T}_\mu)$  and also that

$$\ell_x(\mathcal{T}_0) < \ell_x(\mathcal{T}_\mu) - (2^\mu - 1) \eta \ell_{r+1}. \quad (\text{IV.3})$$

One of the following two cases arises:

- If  $\mathcal{T}_\mu = \mathcal{T}_{\mu+1}$ , then we have that

$$\ell_x(\lambda(\mathcal{T}_0)) = \ell_x(\lambda^{2^\mu}(\mathcal{T}_\mu)) = \ell_x(\lambda^{2^\mu}(\mathcal{T}_{\mu+1})) \geq \ell_x(\mathcal{T}_{\mu+1}) - 2^\mu \eta \ell_{r+1}.$$

Combined with (IV.3) it follows that  $\ell_x(\lambda(\mathcal{T}_0)) > \ell_x(\mathcal{T}_0) - \eta \ell_{r+1}$ , and hence that  $\mathcal{T}_0$  is  $\eta$ -reduced.

- Otherwise, if  $\mathcal{T}_\mu$  is the normalization of  $\lambda^{-2^\mu}(\mathcal{T}_{\mu+1})$ , then we have that  $\ell_x(\mathcal{T}_\mu) < \ell_x(\mathcal{T}_{\mu+1}) - 2^\mu \eta \ell_{r+1}$ , so that

$$\ell_x(\mathcal{T}_0) \leq \ell_x(\mathcal{T}_{\mu+1}) - (2^{\mu+1} - 1) \eta \ell_{r+1}.$$

Since  $\mathcal{T}_{\mu+1}$  is the normalization of  $\lambda(\mathcal{T}_0)$ , we deduce that

$$\ell_x(\lambda(\mathcal{T}_0)) = \ell_x(\mathcal{T}_{\mu+1}) \geq \ell_x(\mathcal{T}_0) + (2^{\mu+1} - 1) \eta \ell_{r+1},$$

whence that  $\mathcal{T}_0$  is  $\eta$ -reduced.

Finally, the last case for when  $\mathcal{T}_0$  is the normalization of  $\lambda^{-1}(\mathcal{T}_1)$  can be treated in the same way.  $\square$

**Remark IV.19.** At each step of the algorithm, the current degree, say  $\tilde{d}_Y$ , in  $Y$ , is smaller than the initial  $d_Y$ . Therefore whenever one chooses  $\eta < \frac{1}{d_Y}$  from the outset, the reduction is at least  $\left\lceil \frac{\tilde{d}_Y}{d_Y} \right\rceil = 1$ . Therefore the algorithm behaves the same as with  $\eta = 0$  and Proposition IV.18 is still valid if  $\eta = 0$ . However, there is no uniform  $\eta > 0$  such that, for every input, the behaviour of the algorithm is the same as with  $\eta = 0$ .

## 4.2 Proof of Theorem IV.2

We prove a slightly more precise theorem than Theorem IV.2, which contains bounds on the bit-size of the entries of the affine map  $U$ :

**Theorem IV.20.** *For any normalized finite subset  $\mathcal{S}$  of  $\mathbb{Z}^2$ , of cardinality  $\sigma$ , convex size  $\pi$ , bounding rectangle  $[0, d_X] \times [0, d_Y]$ , and dense size  $\delta = (d_X + 1)(d_Y + 1)$ , one can compute an invertible affine map  $U \in \text{Aff}(\mathbb{Z}^2)$  as in (IV.1), with  $O(\sigma \log^2 \delta)$  bit-operations, such that:*

- $|\alpha|$ ,  $|\beta|$ ,  $|\alpha'|$ , and  $|\beta'|$  are at most  $\max(2 \max(d_X, d_Y)^2, 1)$ ,
- $|\gamma|$  and  $|\gamma'|$  are at most  $4 \max(d_X, d_Y)^3$ ,
- $U(\mathcal{S})$  is normalized of dense size at most  $9\pi$ .

**Proof.** Proposition IV.4 already covers the degenerate case. In the nondegenerate situation, the theorem follows from Proposition IV.16 for the dense size of the output, from Proposition IV.18 with taking  $\eta = 1/4$  for the bit-complexity, and from Lemma IV.8 for the size of the entries of  $U$ .  $\square$

## 4.3 Timings

We report on performances obtained with our implementation in MAPLE 14 for computing the irreducible factorization of the following polynomials in  $\mathbb{Q}[X, Y]$ :

$$P_n = \left( X^{n+1} + \sum_{i=0}^n i X^i Y^{n-i} \right) \left( Y^{n+1} + \sum_{i=0}^n (n-i) X^i Y^{n-i} \right) \\ \times \left( X^{\lfloor n/2 \rfloor - 1} Y^{\lfloor n/2 \rfloor - 1} + \sum_{i=0}^n X^i Y^{n-i} \right).$$

The source code is available from

<http://www.lix.polytechnique.fr/~berthomieu/convex-dense.htm>.

In Table IV.1, we display timings, in seconds obtained using an INTEL XEON X5450 at 3.0 GHz running LINUX. The first line contains the time spent in the direct call of the native function `factor`. The second line concerns the time spent in our Algorithm IV.1 with  $\eta = 0$ . The last line corresponds to calling `factor` on the reduced polynomial. Indeed, as an optimization, Algorithm IV.1 is run on the set of vertices of the convex hull of the support of the input polynomial. For a set of size  $\sigma$ , it is classical that the convex hull can be computed in time softly linear in  $\sigma$ : we refer the reader to [PS85, Chapter 3, Theorem 3.7] for instance.

$n$	8	16	32	64	128
dense factorization	0.04	0.25	2.3	48	1100
reduction	0.06	0.14	0.28	0.54	1.1
convex factorization	0.04	0.06	0.22	1.5	25

**Table IV.1.** Factorization of  $P_n$ , in seconds.

As expected, our reduction strategy leads to a significant speedup. In fact, with this family, notice that the dense size grows with  $n^2$  while the convex size only grows with  $n$ . We have also tried Algorithm IV.2: the gains are not substantial since most of the time is spent in the factorization. Finally, let us mention that one could investigate the design of a reduction algorithm featuring a dichotomy in the size of the exponents, in a way similar to the half-g.c.d. algorithm (see for instance [GG03, Chapter 11]). This would probably lead to a bit-complexity bound in  $\tilde{O}(\sigma \log \delta)$ . However, the practical impact would be minor as long as the sizes of the exponents are intended to fit into one machine word.

#### 4.4 Optimality of the reduction

It is natural to ask if our algorithm computes the best transformation  $U$  of  $\mathbb{Z}^2$ , that maximizes the ratio of the volumes of  $U(\mathcal{S})$  and  $\mathcal{R}(U(\mathcal{S}))$ , where  $\mathcal{R}(U(\mathcal{S}))$  represents the bounding rectangle of  $U(\mathcal{S})$ .

First, let us mention that the transformations  $\lambda$ ,  $\mu$  and  $\tau_1$  used within our algorithm actually generate  $\text{Aff}(\mathbb{Z}^2)$ . In fact, it is classical that  $\text{SL}(\mathbb{Z}^2)$  is generated by  $\lambda$  and the rotation  $\rho = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$  by the angle  $\pi/2$  [Ser73, Chapter 7, Theorem 2]. Since  $\rho$  can be decomposed into  $\rho = \mu \lambda \mu \lambda^{-1} \mu \lambda \mu$ , and since  $\det \mu = -1$ , we deduce that  $\lambda$  and  $\mu$  generate  $\text{GL}(\mathbb{Z}^2)$ . However, we will not prove that our algorithm returns the best  $U \in \text{Aff}(\mathbb{Z}^2)$  on all input. Roughly speaking, we will only prove that the bound  $3/8$  of the ratio of the volumes at the end of our reduction algorithm is the best bound one can expect in general when  $\eta = 0$ . This bound is attained with the example of Figure IV.10. Specifically, we aim at proving there is no transformation  $U$  such that for all finite subset  $\mathcal{S} \subset \mathbb{Z}^2$ , the inequality  $\text{Vol } U(\mathcal{S}) \geq \alpha \text{Vol } \mathcal{R}(U(\mathcal{S}))$  holds with  $\alpha > 3/8$ :

**Proposition IV.21.** *With the convention  $\frac{\text{Vol } U(\mathcal{S})}{\text{Vol } \mathcal{R}(U(\mathcal{S}))} = 1$  whenever  $\text{Vol } \mathcal{S} = 0$ , one has*

$$\inf_{\mathcal{S} \subset \mathbb{Z}^2, |\mathcal{S}| < \infty} \sup_{U \in \text{Aff}(\mathbb{Z}^2)} \frac{\text{Vol } U(\mathcal{S})}{\text{Vol } \mathcal{R}(U(\mathcal{S}))} = \frac{3}{8},$$

where  $\mathcal{R}(U(\mathcal{S}))$  represents the bounding rectangle of  $U(\mathcal{S})$ .

**Proof.** The degenerate case, that is when  $\text{Vol } \mathcal{S} = 0$ , follows from Proposition IV.4, so that from now on, we can assume that  $\text{Vol } \mathcal{S} > 0$ . By Theorem IV.10, there exists  $U \in \text{Aff}(\mathbb{Z}^2)$  such that  $\text{Vol } U(\mathcal{S}) \geq \frac{3}{8} \text{Vol } \mathcal{R}(U(\mathcal{S}))$ , whence

$$\inf_{\mathcal{S} \subset \mathbb{Z}^2, |\mathcal{S}| < \infty} \sup_{U \in \text{Aff}(\mathbb{Z}^2)} \frac{\text{Vol } U(\mathcal{S})}{\text{Vol } \mathcal{R}(U(\mathcal{S}))} \geq \frac{3}{8}.$$

We shall show that  $\sup_{U \in \text{Aff}(\mathbb{Z}^2)} \frac{\text{Vol } U(\mathcal{S})}{\text{Vol } \mathcal{R}(U(\mathcal{S}))} = \frac{3}{8}$  holds for when  $\mathcal{S} = \{(1, 0), (0, 1), (2, 2)\}$ , which will conclude the proof.

For the rest of the proof,  $\mathcal{S}$  represents the particular set of points  $\{(1, 0), (0, 1), (2, 2)\}$ . As  $\text{Vol } U(\mathcal{S})$  is constant, and equals  $3/2$  for all  $U$ , it suffices to show that, for any  $U \in \text{Aff}(\mathbb{Z}^2)$ ,  $\text{Vol } \mathcal{R}(U(\mathcal{S})) \geq 4$ . As translating and swapping  $X$  and  $Y$  do not change  $\text{Vol } \mathcal{R}(U(\mathcal{S}))$ , we can assume that  $U \in \text{SL}(\mathbb{Z}^2)$ . Let  $\begin{pmatrix} \alpha & \beta \\ \alpha' & \beta' \end{pmatrix}$  be the matricial representation of  $U$ , with  $\alpha \beta' - \beta \alpha' = 1$ . Let  $\rho$  be the rotation by the angle  $\pi/2$ . As  $\text{Vol } \mathcal{R}(\rho(U(\mathcal{S}))) = \text{Vol } \mathcal{R}(U(\mathcal{S}))$ , one can apply  $\rho$ , or  $\rho^{-1}$ , once or twice so that we can further assume that  $\alpha \geq 1$  and  $\alpha' \geq 0$  hold.

If  $\alpha' = 0$ , then  $\alpha \beta' = 1$  so that  $\alpha = 1$  and  $\beta' = 1$ . Since the image of  $(2, 2)$  is  $(2 + 2\beta, 2)$ , the height of the bounding rectangle of  $U(\mathcal{S})$  is 2, and  $\text{Vol } \mathcal{R}(U(\mathcal{S})) \geq 4$  as soon as the horizontal length of  $\mathcal{R}(U(\mathcal{S}))$  is greater or equal to 2. In fact, this length is the maximum of  $|\beta - 1|$ ,  $|2\beta + 1|$  and  $|\beta + 2|$ . If  $|\beta - 1| = 0$ , then  $\beta = 1$  and  $2\beta + 1 = 3$ . Otherwise, if  $|\beta - 1| = 1$ , then either  $\beta = 0$  and  $2\beta + 1 = 1$ , or  $\beta = 2$  and  $2\beta + 1 = 5$ . In this way we observe that, in all cases, the length is at least 2. We can now restrict to considering  $\alpha' \geq 1$ .

If  $\beta = 0$ , then  $\alpha = 1$  and  $\beta' = 1$ . The horizontal length of  $U(\mathcal{S})$  is 2 and its height is  $2\alpha' + 1$ . Therefore we have again that  $\text{Vol } \mathcal{R}(U(\mathcal{S})) \geq 4$ . Similarly, when  $\beta' = 0$ , we have  $\beta = -1$  and  $\alpha' = 1$ : the height of  $U(\mathcal{S})$  is 2 and its horizontal length is at least  $\alpha + 1 \geq 2$ , which yields the same conclusion. Thus, we can now further restrict to considering the case that none of the coefficients of the matrix of  $U$  is zero.

From  $U(1, 0) - U(0, 1) = (\alpha - \beta, \alpha' - \beta')$ , we have  $\text{Vol } \mathcal{R}(U(\mathcal{S})) \geq |\alpha - \beta| |\alpha' - \beta'|$ . Whenever  $|\alpha - \beta| \geq 2$  and  $|\alpha' - \beta'| \geq 2$ , we are done. Therefore, it remains to examine the following cases:

- If  $\alpha = \beta$ , then  $\alpha \beta' - \beta \alpha' = \alpha(\beta' - \alpha') = 1$  implies  $\alpha = \beta = 1$  and  $\beta' = \alpha' + 1$ . A direct calculation yields  $\text{Vol } \mathcal{R}(U(\mathcal{S})) = 3(3\alpha' + 2) \geq 4$ .
- If  $\alpha' = \beta'$ , then  $\alpha \beta' - \beta \alpha' = \alpha'(\alpha - \beta) = 1$  implies  $\alpha' = \beta' = 1$  and  $\alpha = \beta + 1$ , and then  $\text{Vol } \mathcal{R}(U(\mathcal{S})) = 3(3\beta + 2) \geq 4$ , since  $\beta \geq 1$  holds in this case.
- If  $|\alpha - \beta| = 1$ , then we distinguish:
  - if  $\beta = \alpha + 1$ , then the horizontal length of  $\mathcal{R}(U(\mathcal{S}))$  is at least  $3\alpha + 2 \geq 5$ ,
  - if  $\alpha = \beta + 1$ , then the horizontal length of  $\mathcal{R}(U(\mathcal{S}))$  is at least  $3\beta + 2 \geq 5$ .
- If  $|\alpha' - \beta'| = 1$ , then we distinguish:
  - if  $\beta' = \alpha' + 1$ , then the height of  $\mathcal{R}(U(\mathcal{S}))$  is at least  $3\alpha' + 2 \geq 5$ ,
  - if  $\alpha' = \beta' + 1$ , then the height of  $\mathcal{R}(U(\mathcal{S}))$  is at least  $3\beta' + 2 \geq 5$ .  $\square$

## Bibliography

- [AGL04] F. Abu Salem, S. Gao and A. G. B. Lauder. Factoring polynomials via polytopes. In *ISSAC '04: Proceedings of the 2004 International Symposium on Symbolic and Algebraic Computation*, pages 4–11. New York, 2004. ACM.

- [AKS07] M. Avendaño, T. Krick and M. Sombra. Factoring bivariate sparse (lacunary) polynomials. *J. Complexity*, 23(2):193–216, 2007.
- [BCS97] P. Bürgisser, M. Clausen and M. A. Shokrollahi. *Algebraic complexity theory*, volume 315 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, Berlin, 1997.
- [Ber97] L. Bernardin. On square-free factorization of multivariate polynomials over a finite field. *Theoret. Comput. Sci.*, 187(1-2):105–116, 1997. Computer algebra (Saint-Louis, 1996).
- [Ber98] L. Bernardin. On bivariate Hensel lifting and its parallelization. In *ISSAC '98: Proceedings of the 1998 International Symposium on Symbolic and Algebraic Computation*, pages 96–100. New York, 1998. ACM.
- [BHKS09] K. Belabas, M. van Hoeij, J. Klüners and A. Steel. Factoring polynomials over global fields. *J. Théor. Nombres Bordeaux*, 21(1):15–39, 2009.
- [BL10] J. Berthomieu and G. Lecerf. Reduction of bivariate polynomials from convex-dense to dense, with application to factorization. Manuscript to appear in *Math. Comp.*, 2010.
- [BLS+04] A. Bostan, G. Lecerf, B. Salvy, É. Schost and B. Wiebelt. Complexity issues in bivariate polynomial factorization. In *ISSAC '04: Proceedings of the 2004 International Symposium on Symbolic and Algebraic Computation*, pages 42–49. New York, 2004. ACM.
- [Chè10] G. Chèze. A recombination algorithm for the decomposition of multivariate rational functions. Manuscript available from <http://hal.archives-ouvertes.fr/hal-00531601>, 2010.
- [Cox69] H. S. M. Coxeter. *Introduction to Geometry*. John Wiley & Sons Inc., New York, Second edition, 1969.
- [Gao03] S. Gao. Factoring multivariate polynomials via partial differential equations. *Math. Comp.*, 72:801–822, 2003.
- [Gat83] J. von zur Gathen. Factoring sparse multivariate polynomials. In *24th Annual IEEE Symposium on Foundations of Computer Science*, pages 172–179. Los Alamitos, CA, USA, 1983. IEEE Computer Society.
- [GG03] J. von zur Gathen and J. Gerhard. *Modern computer algebra*. Cambridge University Press, Cambridge, Second edition, 2003.
- [GK85] J. von zur Gathen and E. Kaltofen. Factoring sparse multivariate polynomials. *J. Comput. System Sci.*, 31(2):265–287, 1985. Special issue: Twenty-fourth annual symposium on the foundations of computer science (Tucson, Ariz., 1983).
- [GR03] S. Gao and V. M. Rodrigues. Irreducibility of polynomials modulo  $p$  via Newton polytopes. *J. Number Theory*, 101(1):32–47, 2003.
- [GS93] B. Grünbaum and G. C. Shephard. Pick's theorem. *Amer. Math. Monthly*, 100(2):150–161, 1993.
- [HL10] J. van der Hoeven and G. Lecerf. On the bit-complexity of sparse polynomial and series multiplication. Manuscript available from <http://hal.archives-ouvertes.fr/hal-00476223>, 2010.
- [Kal85] E. Kaltofen. Sparse Hensel lifting. In *EUROCAL '85, Vol. 2 (Linz, 1985)*, volume 204 of *Lecture Notes in Comput. Sci.*, pages 4–17. Springer-Verlag, 1985.
- [Kal89] E. Kaltofen. Factorization of polynomials given by straight-line programs. In S. Micali, editor, *Randomness and Computation*, volume 5 of *Advances in Computing Research*, pages 375–412. JAI, 1989.
- [KK06] E. Kaltofen and P. Koiran. Finding small degree factors of multivariate supersparse (lacunary) polynomials over algebraic number fields. In *ISSAC '06: Proceedings of the 2006 International Symposium on Symbolic and Algebraic Computation*, pages 162–168. New York, 2006. ACM.

- [Lec06] G. Lecerf. Sharp precision in Hensel lifting for bivariate polynomial factorization. *Math. Comp.*, 75(254):921–933, 2006.
- [Lec07] G. Lecerf. Improved dense multivariate polynomial factorization algorithms. *J. Symbolic Comput.*, 42(4):477–494, 2007.
- [Lec08] G. Lecerf. Fast separable factorization and applications. *Appl. Algebr. Engrg. Comm. Comput.*, 19(2):135–160, 2008.
- [Lec10] G. Lecerf. New recombination algorithms for bivariate polynomial factorization based on Hensel lifting. *Appl. Algebr. Engrg. Comm. Comput.*, 21(2):151–176, 2010.
- [Ost21] A. M. Ostrowski. Über die Bedeutung der Theorie der konvexen Polyeder für die formale Algebra. *Jahresber. Deutsch. Math.-Verein.*, 30(2):98–99, 1921. Talk given at *Der Deutsche Mathematikertag vom 18–24 September 1921 in Jena*.
- [Ost75] A. M. Ostrowski. On multiplication and factorization of polynomials. I. Lexicographic orderings and extreme aggregates of terms. *Aequationes Math.*, 13(3):201–228, 1975.
- [Ost99] A. M. Ostrowski. On the significance of the theory of convex polyhedra for formal algebra. *SIGSAM Bull.*, 33(1):5, 1999. Translated from [Ost21].
- [Pot08] A. Poteaux. *Calcul de développements de Puiseux et application au calcul du groupe de monodromie d'une courbe algébrique plane*. PhD thesis, Université de Limoges, 2008. Manuscript available from <http://www.lifl.fr/~poteaux>.
- [PR11] A. Poteaux and M. Rybowicz. Complexity bounds for the rational Newton-Puiseux algorithm over finite fields. *Appl. Algebr. Engrg. Comm. Comput.*, 22(3):187–217, 2011.
- [PS85] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Texts and Monographs in Computer Science. Springer-Verlag, New York, 1985. An introduction.
- [Ser73] J.-P. Serre. *A course in arithmetic*. Springer-Verlag, New York, 1973. Translated from the French, Graduate Texts in Mathematics, No. 7.
- [Ste05] A. Steel. Conquering inseparability: primary decomposition and multivariate factorization over algebraic function fields of positive characteristic. *J. Symbolic Comput.*, 40(3):1053–1075, 2005.
- [Wei10] M. Weimann. A lifting and recombination algorithm for rational factorization of sparse polynomials. *J. Complexity*, 26(6):608–628, 2010.
- [Zip79] R. Zippel. Probabilistic algorithms for sparse polynomials. In *EUROSAM '79: Proceedings of the 1979 International Symposium on Symbolic and Algebraic Computation*, volume 72 of *Lecture Notes in Comput. Sci.*, pages 216–226. Berlin, 1979. Springer-Verlag.
- [Zip81] R. Zippel. Newton's iteration and the sparse Hensel algorithm (Extended Abstract). In *SYMSAC '81: Proceedings of the fourth ACM Symposium on Symbolic and Algebraic Computation*, pages 68–72. New York, 1981. ACM.
- [Zip93] R. Zippel. *Effective Polynomial Computation*. Kluwer Academic Publishers, 1993.



# Chapitre V

## Spherical Radon transform and condition number

### Abstract

We study the average complexity of certain numerical algorithms when adapted to solving systems of multivariate polynomial equations whose coefficients belong to some fixed proper real subspace of the space of systems with complex coefficients. A particular motivation is the study of the case of systems of polynomial equations with real coefficients. Along these pages, we accept methods that compute either real or complex solutions of these input systems. This study leads to interesting problems in Integral Geometry: the question of giving estimates on the average of the normalized condition number along great circles that belong to a Schubert subvariety of the Grassmannian of great circles on a sphere. We prove that this average equals a closed formula in terms of the spherical Radon transform of the condition number along a totally geodesic submanifold of the sphere. This chapter is based on an accepted for publication article written with LUIS MIGUEL PARDO [BP11b].

## 1 Introduction

### 1.1 The context of our new results

The main result of these pages is motivated by the study of the real version of Smale's 17th Problem. In [Sma00], S. Smale proposed the following problem:

**Problem V.1. (Smale's 17th Problem)** *“Can a zero of  $n$  complex polynomial equations in  $n$  unknowns be found approximately, on the average, in polynomial time with a uniform algorithm?”*

This problem was answered affirmatively in [BP09b]: The authors exhibited a **ZPP** (Las Vegas) algorithm that solves systems of complex multivariate polynomial equations in average time  $O(N^3)$ , where  $N$  is the input length for dense encoding of multivariate polynomials (*cf.* also [BP09a] for a survey on the topic). Another **ZPP** algorithm solving the same problem in average time  $O(N^2)$  was shown in [BP11a].

There is, however, much room for improvement and further research. Some open questions follow:

- Find a deterministic *average polynomial time* algorithm that solves systems of multivariate complex polynomial equations. Some deep advances in this direction have been made in [BC11]. These authors use the powerful “smoothed analysis”, by Cheng and Spielman, to exhibit a deterministic algorithm with sub-exponential average time with a small exponent of order  $O(\log_2 \log_2 N)$ . But the problem of a deterministic average polynomial time algorithm remains open.
- Find an algorithm (either deterministic or probabilistic) with polynomial complexity on average that solves systems of multivariate polynomial equations when the inputs are given by encoding alternatives to dense encoding: sparse/fewnomial systems, straight-line program encoding, etc... To our knowledge, no meaningful advance has been made to date in this direction.

In his original statement of Problem 17th, S. Smale also addressed the question about real solving:

**Problem V.2. (Smale’s 17th Problem, real case)** “...*Similar, more difficult, problems may be raised for real polynomial systems (and even with inequalities).*”

Namely, try to solve real systems in average polynomial time. In these pages we focus on this real case of Smale’s problem. To date, real solving systems of systems of polynomial equations with real coefficients has shown strong resistance to be solved in polynomial time on average.

There are two main approaches dealing with this kind of problems: Symbolic/Geometric and Numerical Solving. We are not concerned here with Symbolic/Geometric methods, in here. The reader interested in this approach may follow [BGH+12, BGHP05, BGHP09, BGH+10, BPR06] and references therein.

In this chapter, we are concerned with the numerical approach. A serious attempt to solve numerically systems of polynomial equations with real coefficients was made in the series of manuscripts [CKMW08, CKMW09, CKMW10]. Their proposal is based on the study of the probability distribution of a real condition number and then apply exhaustive search. The complexity has not been shown to be tractable.

Other studies of the properties of real systems on average have been made in [BCL06, BP08, McL02] and references therein. Other attempts to use search algorithms (in this case, using exclusive methods) may be found in [DY93] and references therein.

On a completely different basis, a very positive experiment, using evolutive algorithms, is exhibited in [Bor11]: The experiment shows excellent performance and a high probability of success to find an approximate zero for real zeros of real systems of multivariate polynomial equations. However, these experiments lack appropriate mathematical foundations.

Nevertheless, search is not necessarily the unique approach to numerical solving of real systems. Firstly, because we may not be interested in computing *all* solutions (which certainly forces an exponential running time) but computing *one* solution (see [BP06] for a discussion between universal and non-universal solving in numerical analysis). As in the methods shown to be efficient in the complex case, one may try to use an *homotopic deformation* technique approach (also called path following methods or continuation methods) to compute just one (real or complex) solution of systems of real polynomial equations. See, for instance, the books [AG90, BCSS98, Mor87, SW05], or surveys like [BP09a, Li03] and references therein for different statements of the algorithmic scheme of continuation methods.

The main drawback to the use of an homotopic deformation technique for systems with real coefficients is the codimension of the discriminant variety  $\Sigma^{\mathbb{R}}$  in the space of polynomial equations with real coefficients  $\mathcal{H}_{(d)}^{\mathbb{R}}$ . Since the codimension of  $\Sigma^{\mathbb{R}}$  in  $\mathcal{H}_{(d)}^{\mathbb{R}}$  is one and since the number of real solutions (in  $\mathbb{P}_n(\mathbb{R})$ ) is constant along each connected components of  $\mathcal{H}_{(d)}^{\mathbb{R}} \setminus \Sigma^{\mathbb{R}}$ , we conclude:

- The number of connected components outside the discriminant variety is exponential in the number  $(n + 1)$  of variables.
- The probability that for any two randomly chosen systems  $f, g \in \mathcal{H}_{(d)}^{\mathbb{R}}$ , every continuous path joining them in  $\mathcal{H}_{(d)}^{\mathbb{R}}$  intersects the discriminant variety  $\Sigma^{\mathbb{R}}$  is greater than the probability that they have a different number of real solutions (in  $\mathbb{P}_n(\mathbb{R})$ ). To our knowledge there is no precise estimate for this quantity. See some related estimates in [AW05, BP08, SS93b, SS96, Wsc05] and references therein.
- In the case of linear deformations, for any two randomly chosen systems  $f, g \in \mathcal{H}_{(d)}^{\mathbb{R}}$  of norm 1, the expected number of points in the intersection between the great circle joining  $f$  and  $g$  and the discriminant variety equals the (codimension one) volume of the projection of the discriminant variety onto the sphere in  $\mathcal{H}_{(d)}^{\mathbb{R}}$  of radius one. This is a mere consequence of Crofton-Poincaré's formula.

The facts cause some troubles for the standard method based on a lifting of these paths (through a covering map) and force alternatives. One could be the proposal in [BS09]: follow a path inside the solution variety. This method has the inconvenience that there is no known method to construct the path to be followed without prior knowledge of the zero to be computed. This could be, perhaps, improved if we were able to compute geodesics with respect to the non-linear condition number metric (*cf.* the excellent manuscript [BDMS09], for instance). But, for the moment, there is no efficient method to compute them. Another proposal for real systems of equations could be that of [BS11], which traces real curves connecting the solutions of one system of equations to those of another but, in this case, no estimate of the number of steps is provided and, hence, no complexity estimate is known.

A different proposal is the one we suggest in these pages. First we choose to follow simplest paths as in the complex case: great circles on spheres. Then, instead of trying to solve real systems of multivariate polynomial equations by homotopic deformation that follows a path that goes from real systems to real systems, we propose to open up the space and *apply an homotopic deformation by following paths that begin in a complex (not real) initial system of equations and ends in a real system of equations*. This may be modeled in a simple saying:

*Apply the (complex) algorithm described in [BP11a] to real systems of polynomial equations and study its average complexity.*

Certainly this approach is not expected to provide only real solutions of real systems: we just want to know if there is a low average complexity algorithm that computes approximate zeros of a single solution of systems of equations with real coefficients, accepting both real and complex solutions without establishing any preference among them.

This study leads to interesting problems in Integral Geometry, some of which are solved here. In principle, studying the average complexity of this kind of algorithm leads to the question of giving estimates on the average behavior of condition number along great circles that belong to certain Schubert subvariety of the Grassmannian of great circles on a sphere. We prove that this average equals a closed formula in terms of the spherical Radon transform of the condition number along an  $N$ -dimensional totally geodesic submanifold of the sphere of systems of polynomial equations with complex coefficients. This is the main result in these pages.

## 1.2 Statement of the main results

The first result explains the behavior of the expected value of an integrable function in certain Schubert subvarieties of real Grassmannians given as the set of great circles that intersect a given vector subspace. In order to state it we need to introduce some notation.

Let  $S^n \subseteq \mathbb{R}^{n+1}$  be the real hypersphere of radius one centered at the origin. For a real vector subspace  $M \subseteq \mathbb{R}^{n+1}$  we denote by  $S(M) \subseteq S^n$  the hypersphere defined by  $M$ . From now on, we assume that the codimension of  $M$  in  $\mathbb{R}^{n+1}$  is greater than 2. We assume that  $S^n$  is endowed with the standard Riemannian structure and we denote by  $d S^n$  its canonical volume form. We denote by  $d_R$  the Riemannian distance in  $S^n$  and by  $d_{\mathbb{P}}$  the “projective” distance (*i.e.*  $d_{\mathbb{P}}(f, g) = \sin d_R(f, g)$ , for all  $f, g \in S^n$ ). As the total volume of  $S^n$  is finite, we may define a probability distribution on  $S^n$  in the canonical way. Similarly, we may define in  $S(M)$  and  $S^n \times S(M)$  their canonical probability distributions. Given a point  $(g, f) \in S^n \times S(M)$ , we denote by  $L_{(g, f)}$  the great circle in  $S^n$  passing through  $f$  and  $g$ . We may assume on  $L_{(g, f)}$  the standard volume form  $d L_{(g, f)}$  (the standard length). We begin by recalling the definition of spherical Radon Transform from [Rub02].

**Definition V.3. ([Rub02])** *With the same notation, let  $\varphi: S^n \rightarrow \mathbb{R}_+$  be an integrable function, and let  $k = n - p$  be the codimension of  $M$  in  $\mathbb{R}^{n+1}$ . The spherical Radon transform of  $\varphi$  with respect to  $S(M)$  of order  $\alpha$  is defined in the following terms:*

$$R^\alpha \varphi(S(M)) = \rho_{n,p}(\alpha) \int_{S^n} \frac{\varphi(g)}{d_{\mathbb{P}}(g, S(M))^{n-p-\alpha}} d S^n,$$

where

$$\rho_{n,p}(\alpha) = \frac{B\left(\frac{n-p-\alpha+1}{2}, \frac{\alpha+p-1}{2}\right)}{\nu_n},$$

$\nu_n$  is the standard volume of the unit sphere  $S^n$  and  $B$  is the usual Beta function.

**Remark V.4.** Our normalization constant  $\rho_{n,p}(\alpha)$  differs slightly from  $\gamma_{n,p}(\alpha)$ , the one used in [Rub02]. Multiplying by  $\rho_{n,p}(\alpha)^{-1} \gamma_{n,p}(\alpha)$  we obviously obtain the original definition of B. Rubin.

Then, we prove:

**Theorem V.5.** *With the same notation as above, for every integrable function  $\varphi: S^n \rightarrow \mathbb{R}_+$ , let  $E$  be the expectation given by the following identity:*

$$E = E_{(g,f) \in S^n \times S(M)} \left[ \int_{L(g,f)} \varphi(h) \, dL_{(g,f)}(h) \right].$$

Moreover, for every  $n, p$  and  $i$ , let us define the constants:

$$C(n, p, i) := 2 \binom{\frac{n-p}{2}}{i} \frac{B\left(\frac{n+2}{2}, \frac{1}{2}\right)}{B\left(\frac{p-1}{2}, \frac{1}{2}\right)} \text{ and } B_0(n, p, i) := 2 \binom{\frac{n-p-3}{2}}{i} \frac{B\left(\frac{n+2}{2}, \frac{1}{2}\right)}{B\left(\frac{p-1}{2}, \frac{1}{2}\right)}.$$

In terms of the value of the codimension  $k = n - p$ , the following equalities and inequalities hold:

1. If  $k = 1$ , then

$$\frac{4\sqrt{2}\pi}{(n+\sqrt{3})^{1/2}} E_{S^n}[\varphi] \leq E \leq \frac{\sqrt{(n-2)\pi}}{2} \mathbf{R}^0 \varphi(S(M));$$

2. If  $k \in 2\mathbb{N}^*$ , then

$$E = \sum_{i=0}^{\frac{n-p-2}{2}} C(n, p, i) \mathbf{R}^{n-p-2i-1} \varphi(S(M));$$

3. If  $k \in (2\mathbb{N}^* + 1)$ , then

$$E \geq \sum_{i=0}^{\frac{n-p-3}{2}} \frac{(n-2) B_0(n, p, i)}{\sqrt{i + \sqrt{3/2}}} \mathbf{R}^{n-p-2i-2} \varphi(S(M)),$$

$$E \leq \sum_{i=0}^{\frac{n-p-3}{2}} \frac{8 B_0(n, p, i)}{(2i+1)(n-2)} \mathbf{R}^{n-p-2i-2} \varphi(S(M)).$$

**Remark V.6.** Note that using Gautschi's [Gau60] and Kershaw's [Ker83] inequalities we also have the following sharp bounds of our coefficients:

$$2 \binom{\frac{n-p}{2} - 1}{i} \sqrt{\frac{p - \frac{3}{2}}{n + \sqrt{3}}} \leq C(n, p, i) \leq 2 \binom{\frac{n-p}{2} - 1}{i} \sqrt{\frac{p - 3 + \sqrt{3}}{n + \frac{3}{2}}},$$

$$\frac{(n-2) B_0(n, p, i)}{\sqrt{i + \sqrt{3}/2}} \geq 2 \binom{\frac{n-p-3}{2}}{i} (n-2) \sqrt{\frac{(2p-3)}{(2i + \sqrt{3})(n + \sqrt{3})}},$$

and

$$\frac{8 B_0(n, p, i)}{(2i+1)(n-2)} \leq 16 \binom{\frac{n-p-3}{2}}{i} \frac{\sqrt{2(p-3 + \sqrt{3})}}{(2i+1)(n-2)\sqrt{2n+3}}.$$

Note that the largest integral terms in identities (2) and (3) of Theorem V.5 correspond to the case  $i = 0$ . Some less sharp, but illustrative, upper and lower bounds are exhibited in the following corollary.

**Corollary V.7.** *With the same notation as above, for  $k = n - p \geq 2$ ,  $E$  is bounded as follows:*

$$2 \sqrt{\frac{p + \frac{1}{2}}{n + \sqrt{3}}} \mathbf{R}^1 \varphi(S(M)) \leq E \leq 2 \sqrt{\frac{p - 1 + \sqrt{3}}{n + \frac{3}{2}}} \frac{1}{B(\frac{n-p}{2}, \frac{p}{2})} \mathbf{R}^1 \varphi(S(M)).$$

Note that the upper bound satisfies:

$$\frac{1}{B(\frac{n-p}{2}, \frac{p}{2})} \mathbf{R}^1 \varphi(S(M)) = E_{S^n} \left[ \frac{\varphi(g)}{d_{\mathbb{P}}(g, S(M))^{n-p-1}} \right],$$

where  $E_{S^n}$  means expectation.

In the path to the proof of this statement, we also prove the following integral formula in some incidence subvariety of the Grassmann manifold:

Let  $\mathcal{L}$  be the Grassmannian given as the set of great circles in  $S^n$  and denote by  $\mathcal{L}_M$  the semialgebraic subset defined as those great circles  $L \in \mathcal{L}$  such that  $L \cap M \neq \emptyset$ . We shall see that  $\mathcal{L}_M$  may be decomposed as a union of two real manifolds  $\mathfrak{C}_M \cup G_{2,p+1}(\mathbb{R})$ , where  $\mathfrak{C}_M$  is the manifold of all great circles  $L \in \mathcal{L}$  that intersect  $S(M)$  in exactly 2 points and,  $G_{2,p+1}(\mathbb{R})$  is the Grassmannian of great circles in  $S(M)$ . In fact,  $\mathfrak{C}_M$  is formed by smooth regular points of maximal dimension in  $\mathcal{L}_M$  and is a dense semialgebraic subset of  $\mathcal{L}_M$ .

The Riemann manifold  $\mathfrak{C}_M$  is endowed with a natural volume form that we denote by  $d\nu_M$ . This volume form extends to its closure  $\mathcal{L}_M$  as a measure in the obvious way. For every function  $\varphi: \mathcal{L}_M \rightarrow \mathbb{R}$  we denote by

$$\int_{\mathcal{L}_M} \varphi d\nu_M,$$

the integral of the restriction of  $\varphi$  to  $\mathfrak{C}_M$  with respect to  $d\nu_M$  and for every subset  $F \subseteq \mathcal{L}_M$  we denote by  $\nu_M[F]$  the volume of the intersection  $F \cap \mathfrak{C}_M$ . We will prove that the volume  $\nu_M[\mathcal{L}_M]$  is finite and, hence, this induces a natural probability distribution in  $\mathcal{L}_M$ .

Next, for every  $L \in \mathcal{L}_M$ , we have a function  $d_M: L \rightarrow \mathbb{R}_+$  given by  $d_M(h) = d_{\mathbb{P}}(h, S(M))^{\text{codim}_{\mathbb{R}^{n+1}}(M)^{-1}}$ . We may define a measure on every line  $L \in \mathcal{L}_M$  that we denote  $d\nu_M$  given by

$$E_{L_M}[\varphi] = \frac{1}{\text{vol}_M[L]} \int_L \varphi d_M(x) dL,$$

where

$$\text{vol}_M[L] = \int_L d_M(x) dL = B\left(\frac{k+2}{2}, \frac{1}{2}\right) \partial_M(L)^{k-1},$$

where  $k = \text{codim}_{\mathbb{R}^{n+1}}(M)$  and  $\partial_M(L) = \max \{d_{\mathbb{P}}(h, S(M)), h \in L\}$ .

In the path to prove the main result (Theorem V.5) we also prove the following statement:

**Proposition V.8.** *With the same notation as above, for every integrable function  $\varphi: S^n \rightarrow \mathbb{R}_+$  the following equality holds :*

$$E_{\mathcal{L}_M}[E_{L_M}[\varphi]] = E_{S^n}[\varphi].$$

In particular, we have

$$\text{vol}[\mathcal{L}_M] = \frac{\text{vol}[S^n] \text{vol}[S(M)]}{B\left(\frac{k+1}{2}, \frac{1}{2}\right)},$$

where  $k$  is the codimension of  $M$  in  $\mathbb{R}^{n+1}$ .

### 1.3 The case of polynomial equations

As said before, the motivation of this study is the analysis of the average complexity of homotopic deformation algorithms for polynomial system solving. Here we will state some corollaries of Theorem V.5 and of Proposition V.8 above. We need some additional notation to state these corollaries.

For every positive integral number  $d \in \mathbb{N}$ , let  $H_d$  be the complex vector space spanned by the homogeneous polynomials  $f \in \mathbb{C}[X_0, \dots, X_n]$  of degree  $d$ . The complex space  $H_d$  is naturally endowed with a unitarily-invariant Hermitian inner product, known as Bombieri's Hermitian product (other authors use the terms Bombieri-Weyl's or even Kostlan's norm for the associated norm, cf. [BCSS98] for details). For every degree list  $(d) = (d_1, \dots, d_n)$  of positive integer numbers, we denote by  $\mathcal{H}_{(d)}$  the complex vector space given as the product  $\mathcal{H}_{(d)} = \prod_{i=1}^n H_{d_i}$ . Note that if for every  $i$ ,  $1 \leq i \leq n$ ,  $f_i \in \mathbb{C}[X_0, \dots, X_n]$  is homogeneous of degree  $d_i$ , then  $\mathcal{H}_{(d)}$  may be seen as the vector space of homogeneous systems of equations  $f = (f_1, \dots, f_n)$ . The complex space  $\mathcal{H}_{(d)}$  is endowed with the unitarily-invariant Hermitian product  $\langle \cdot, \cdot \rangle_{\Delta}$  defined as the Cartesian product of Bombieri's Hermitian products in  $H_{d_i}$ .

Let us denote by  $N+1$  the complex dimension of  $\mathcal{H}_{(d)}$  and by  $\mathcal{D} = \prod_{i=1}^n d_i$  the *Bézout number* associated to the list  $(d) = (d_1, \dots, d_n)$ .

Let  $\|\cdot\|_{\Delta}$  be the norm associated to  $\langle \cdot, \cdot \rangle_{\Delta}$  and let us denote by  $\mathbb{S}^{2N+1} = \mathbb{S}(\mathcal{H}_{(d)})$  the unit sphere in  $\mathcal{H}_{(d)}$  with respect to the norm  $\|\cdot\|_{\Delta}$ .

For every systems of equations  $f = (f_1, \dots, f_n) \in \mathcal{H}_{(d)}$ , we denote by  $V_{\mathbb{P}}(f) \subseteq \mathbb{P}^n(\mathbb{C})$  the complex projective algebraic variety of their common zeros. Namely,

$$V_{\mathbb{P}}(f) = \{\zeta \in \mathbb{P}^n(\mathbb{C}), f_i(\zeta) = 0, 1 \leq i \leq n\}.$$

Given  $f \in \mathcal{H}_{(d)}$  and given  $\zeta \in V_{\mathbb{P}}(f)$ , we denote by  $\mu_{\text{norm}}(f, \zeta)$  the normalized condition number of  $f$  at  $\zeta$  (as introduced in [SS93a]) and for every positive real  $\alpha \in \mathbb{R}$ , we will denote by  $\mu_{\text{av}}^{\alpha}(f)$  the average of the  $\alpha$ th power of condition number of  $f$  along its complex zeros. Namely,

$$\mu_{\text{av}}^{\alpha}(f) = \frac{1}{\sharp(V_{\mathbb{P}}(f))} \sum_{\zeta \in V_{\mathbb{P}}(f)} \mu_{\text{norm}}^{\alpha}(f, \zeta).$$

Studies of the average values of  $\mu_{\text{av}}(f)^{\alpha}$ , for  $1 \leq \alpha < 4$  are exhibited in [BP11a].

From [Shu09] (and the explicit descriptions of the constants in [BC11, Bell1, DMS11]), the number of deformation homotopy steps along a great circle path performed by Newton's method from an initial system  $g$  with initial zero  $\zeta \in V_{\mathbb{P}}(g)$  and target system  $f$  is bounded by the quantity:

$$\mathcal{C}(f, g, \zeta) = \int_L \mu_{\text{norm}}(h, \zeta_h)^2 dL,$$

where  $L$  is the great circle containing  $g$  and  $f$  (which is assumed not to intersect the discriminant variety  $\Sigma \subseteq \mathbb{S}(\mathcal{H}_{(d)})$ ).

Now we consider a probabilistic (we see it is **Zero-Error Probability** or, in fact, **Las Vegas** in our case) algorithm based on the one introduced in [BP11a], with set of initial pairs  $\mathcal{G}_{(d)}$  that we call **BP** in the sequel. We also consider  $M \subseteq \mathcal{H}_{(d)}$  a real vector subspace of the space of complex systems. For instance,  $M$  can be the real vector subspace  $\mathcal{H}_{(d)}^{\mathbb{R}}$  of  $\mathcal{H}_{(d)}$  of systems of equations with real coefficients. Another example could be the sparse case defined by the real vector space of polynomials with coefficients in a given polytope.

We denote by  $\mathbb{S}(M) \subseteq \mathbb{S}^{2N+1}$  the sphere of radius 1 given by points in  $M$  with respect to Bombieri-Weyl's norm.

Our goal is the design of algorithms adapted to  $M$  as input space. Our proposal here will be the following variation of **BP**:

#### Algorithm V.1

##### Variant of **BP** algorithm for real systems

**Input.** A system  $f \in M$ .

**Output.** Either **FAILURE** or an approximate zero  $z \in \mathbb{P}^n(\mathbb{C})$  of  $f$  with associated zero  $\zeta \in \mathbb{P}^n(\mathbb{C})$ .

1. Guess at random  $(g, \zeta) \in \mathcal{G}_{(d)}$ .
2. Apply deformation homotopy with initial pair  $(g, \zeta)$  and target  $f$ .

The first obvious consequence of our study is the following one:

**Corollary V.9.** *Let  $\Sigma \subseteq \mathbb{S}^{2N+1}$  be the discriminant variety (as defined in [BCSS98, SS93a]). Assume that  $\dim(\Sigma \cap \mathbb{S}(M)) < \dim \mathbb{S}(M)$ . Then, the probability that the algorithm above outputs **FAILURE** is 0. Namely, the probability that the algorithm outputs an approximate zero associated to some input system  $f \in M = \mathcal{H}_{(d)}^{\mathbb{R}}$  is 1.*

Nevertheless, the problem is not the soundness of the algorithm, but the average complexity. The usual upper bound for the average complexity of such an algorithm (assuming Gaussian distribution on  $M$ ) will be the expected value

$$E_M = E_M[\text{Time}] = E_{\mathcal{G}(d) \times \mathbb{S}(M)}[\mathcal{C}(f, g, \zeta)].$$

The following statements are different estimates for this quantity  $E$ .

As in the previous subsection, we will denote by  $\mathcal{L}$  the Grassmannian of real great circles in  $\mathbb{S}^{2N+1}$  and by  $\mathcal{L}_M$  the great circles in  $\mathcal{L}$  that intersect  $\mathbb{S}(M)$ .

From Theorem V.5 we also obtain the following consequence:

**Corollary V.10.** *With the same notation as above, assume  $\dim M = p + 1$  and let us  $C(2N + 1, p, i)$ ,  $B_0(2N + 1, p, i)$  be the same constants as defined in Theorem V.5. Let  $k = 2N - p + 1$  be the codimension, then, we have*

1. If  $k = 1$ , then:

$$\frac{4\sqrt{\pi}}{(N+2)^{1/2}} E_{\mathbb{S}^{2N+1}}[\mu_{\text{av}}^2] \leq E \leq \sqrt{\frac{(N - \frac{1}{2})\pi}{2}} \mathbf{R}^0[\mu_{\text{av}}^2](\mathbb{S}(M));$$

2. If  $k \in 2\mathbb{N}^*$ , then the average estimate of the complexity based on the condition number  $E_M$  satisfies:

$$E_M = \sum_{i=0}^{\frac{2N-p-1}{2}} C(2N+1, p, i) \mathbf{R}^{2(N-i)-p}[\mu_{\text{av}}^2](\mathbb{S}(M));$$

3. If  $k \in (2\mathbb{N}^* + 1)$ , then

$$E_M \geq \sum_{i=0}^{\frac{2N-p-2}{2}} \frac{(2N-1) B_0(2N+1, p, i)}{\sqrt{i + \sqrt{3/2}}} \mathbf{R}^{2(N-i)-p-1}[\mu_{\text{av}}^2](\mathbb{S}(M)), \quad (\text{V.1})$$

$$E_M \leq \sum_{i=0}^{\frac{2N-p-2}{2}} \frac{8 B_0(2N+1, p, i)}{(2i+1)(2N-1)} \mathbf{R}^{2(N-i)-p-1}[\mu_{\text{av}}^2](\mathbb{S}(M)).$$

We also have:

**Corollary V.11.** *With the same notation as above, the following inequalities hold:*

$$2\sqrt{\frac{p + \frac{1}{2}}{2N+1 + \sqrt{3}}} \mathbf{R}^1[\mu_{\text{av}}^2](\mathbb{S}(M)) \leq E_M \leq 2\sqrt{\frac{p-1 + \sqrt{3}}{2N + \frac{5}{2}}} \left( \frac{\mathbf{R}^1[\mu_{\text{av}}^2](\mathbb{S}(M))}{\mathbf{B}(N + \frac{1-p}{2}, \frac{p}{2})} \right).$$

Or, equivalently,

$$\sqrt{\frac{2p + \frac{1}{2}}{2N+1 + \sqrt{3}}} \mathbf{R}^1[\mu_{\text{av}}^2](\mathbb{S}(M)) \leq E_M,$$

$$E_M \leq 2\sqrt{\frac{p-1 + \sqrt{3}}{2N + \frac{5}{2}}} E_{\mathbb{S}^{2N+1}} \left[ \frac{\mu_{\text{av}}^2(g)}{d_{\mathbb{P}}(g, \mathbb{S}(M))^{2N-p}} \right].$$

**Corollary V.12.** *With the same notation as above, let  $p+1$  be the dimension of  $M$  and  $k = 2N - p + 1$  be the codimension of  $M$  in  $\mathcal{H}_{(d)}$ . Then the following equality holds:*

$$E_M = T(N, p) E_{\mathcal{L}_M} \left[ \frac{1}{\partial_M(L)} \int_L \mu_{\text{av}}^2(h) \, dL \right],$$

where

$$T(N, p) = \frac{2 \, \text{B}(N + \frac{3}{2}, \frac{1}{2})}{\text{B}(N + 1 - \frac{p}{2}, \frac{1}{2})}.$$

Note that, according to Gautschi's and Kershaw's bounds,  $T(N, p)$  is asymptotically in  $\Theta\left(1 - \frac{p}{2N}\right)^{1/2}$ .

Now we are in conditions to exhibit some average complexity upper bounds for the application of the algorithm in [BP11a] to systems with real coefficients. This is resumed in the following corollary.

**Corollary V.13.** *Assume now that  $M$  is the real vector subspace of systems with real coefficients (i.e.  $M = \mathcal{H}_{(d)}^{\mathbb{R}}$ ). Denote by  $E_{\mathbb{R}}$  the expected number of steps of the underlying homotopy of [Shu09] (i.e.  $E_{\mathbb{R}} = E_M$  under our hypothesis). As  $\dim_{\mathbb{R}} M = p = N + 1$  and  $\dim_{\mathbb{R}} \mathcal{H}_{(d)} = 2N + 2$ , then the codimension  $k$  of  $M$  is  $N + 1$  and the following holds:*

1. *If the codimension  $(N + 1) \in 2\mathbb{N}^*$ , then  $E_{\mathbb{R}}$  satisfies:*

$$E_{\mathbb{R}} = \sum_{i=0}^{\frac{N-1}{2}} C(2N + 1, N, i) \mathbf{R}^{N-2i} [\mu_{\text{av}}^2](\mathbb{S}(\mathcal{H}_{(d)}^{\mathbb{R}})),$$

2. *If the codimension  $(N + 1) \in (2\mathbb{N}^* + 1)$ , then  $E_{\mathbb{R}}$  satisfies the following inequalities:*

$$2 \sqrt{\frac{N + \frac{1}{2}}{2N + 1 + \sqrt{3}}} \mathbf{R}^1[\mu_{\text{av}}^2](\mathbb{S}^N) \leq E_{\mathbb{R}} \leq 2 \sqrt{\frac{N - 1 + \sqrt{3}}{2N + \frac{5}{2}}} E_{\mathbb{S}^{2N+1}} \left[ \frac{\mu_{\text{av}}^2(g)}{d_{\mathbb{P}}(g, \mathbb{S}^N)^N} \right],$$

and therefore

$$\mathbf{R}^1[\mu_{\text{av}}^2](\mathbb{S}^N) \leq E_{\mathbb{R}} \leq \sqrt{2} \left( \frac{\mathbf{R}^1[\mu_{\text{av}}^2](\mathbb{S}^N)}{\text{B}(\frac{N+1}{2}, \frac{N}{2})} \right) = \sqrt{2} E_{\mathbb{S}^{2N+1}} \left[ \frac{\mu_{\text{av}}^2(g)}{d_{\mathbb{P}}(g, \mathbb{S}^N)^N} \right],$$

where  $\mathbb{S}^N = \mathbb{S}(\mathcal{H}_{(d)}^{\mathbb{R}})$  and  $\mathbb{S}^{2N+1} = \mathbb{S}(\mathcal{H}_{(d)})$ .

The manuscript is structured as follows. In Section 2 we establish some basic facts about the underlying geometry of  $\mathcal{L}_M$  as semialgebraic set and we also describe the Riemannian structure at regular points. In Section 3 we prove some technical results from Integral Geometry (mostly computing some normal Jacobians and basic integrals). In Section 4 we prove Theorem V.5, Corollary V.7 and Proposition V.8 (the results stated in Section 1.2 above). In Section 5 we prove the corollaries stated in Section 1.3 above.

## Acknowledgments

We wish to thank Michael Shub for his suggestion to rewrite our results in terms of the spherical Radon transform of B. Rubin, and both anonymous referees for their several helpful comments.

## 2 The underlying geometry

The aim of this section is to prove the following statement concerning the geometry of the Schubert variety (as semialgebraic set)  $\mathcal{L}_M$ . We have not found an appropriate reference where both the algebraic geometry and the Riemannian metric statements (including an explicit description of the tangent spaces to the smooth points of  $\mathcal{L}_M$ ) of the following lemma are stated. As we need both of them to prove our Theorem V.5, we decided to include a self-contained proof.

**Lemma V.14.** *Let  $M \subseteq \mathbb{R}^{n+1}$  be a proper vector subspace of dimension  $p + 1$  and codimension  $k = n - p > 0$ . Let  $\mathcal{L}_M$  be the set of great circles in  $\mathcal{L}$  such that  $L \cap S(M) \neq \emptyset$ . Then, the following properties hold:*

1. *The semialgebraic set  $\mathcal{L}_M$  decomposes as the union of two Riemannian manifolds  $\mathfrak{C}_M \cup G_{2,p+1}(M)$ , where*
  - *$\mathfrak{C}_M$  is the set of great circles  $L \in \mathcal{L}$  such that  $L$  intersects  $S(M)$  in exactly two points (i.e.  $\sharp(L \cap S(M)) = 2$ ),*
  - *$G_{2,p+1}(M)$  may be identified with the Grassmannian of great circles in  $S(M)$ .*
2. *Manifold  $\mathfrak{C}_M$  is made of smooth regular points of maximal dimension in  $\mathcal{L}_M$  and it is a dense subset of  $\mathcal{L}_M$  with respect to the topology induced in  $\mathcal{L}_M$  by the Riemannian metric of  $\mathcal{L}$ .*
3. *The dimension of  $\mathfrak{C}_M$  equals the dimension of  $\mathcal{L}_M$  and satisfies:*

$$\dim_{\mathbb{R}} \mathfrak{C}_M = \dim_{\mathbb{R}} \mathcal{L}_M = n + p - 1.$$

4. *For every great circle  $L \in \mathfrak{C}_M$  given as the intersection with  $S(M)$  of a real plane spanned by a matrix  $A$  in the Stiefel manifold  $ST_{2,n+1}(\mathbb{R})$ , the tangent space  $T_L \mathfrak{C}_M$  can be isometrically identified with*

$$T_L \mathfrak{C}_M = \{B \in T_L G_{2,n+1}(\mathbb{R}), \exists \eta \in T_f S^p, (\text{Id}_{n+1} - A^T A) \eta^T = B^T A f^T\},$$

*where  $\{\pm f\} = L \cap S(M)$ ,  $G_{2,n+1}(\mathbb{R})$  is the Grassmannian of great circles in  $S^n$ ,  $A^T$ ,  $\eta^T$ ,  $B^T$ ,  $f^T$  are respectively the transposed matrices of  $A$ ,  $\eta$ ,  $B$ ,  $f$  and  $\text{Id}_{n+1}$  is the  $(n + 1) \times (n + 1)$  identity matrix.*

### 2.1 Some known facts about Grassmannian, Schubert and incidence varieties

We have not found any appropriate reference for the details of this statement, hence we prove it here. Firstly, we just identify  $M = \mathbb{R}^{p+1}$  and  $S(M) = S^p$  and prove the lemma for this particular case.

We denote by  $\mathcal{L}_n = G_{2,n+1}(\mathbb{R})$  (or simply  $\mathcal{L}$  when no confusion arises) the Grassmannian of great circles in  $S^n$ . Recall that the Stiefel manifold  $\text{ST}_{2,n+1}(\mathbb{R})$  is the real manifold of dimension  $2n - 1$  whose points are orthonormal bases of planes in  $\mathbb{R}^{n+1}$  (written as  $2 \times (n+1)$  matrices). For every matrix  $A \in \text{ST}_{2,n+1}(\mathbb{R})$  the tangent space  $T_A \text{ST}_{2,n+1}(\mathbb{R})$  is given by the following identity:

$$T_A \text{ST}_{2,n+1}(\mathbb{R}) := \{B \in \mathcal{M}_{2,n+1}(\mathbb{R}), BA^T + AB^T = 0\},$$

where  $A^T$  still means transpose. For the remainder of this section we simplify notation by writing  $\text{ST}(\mathbb{R}) = \text{ST}_{2,n+1}(\mathbb{R})$ .

There is a natural left action defined by  $O(2)$  over  $\text{ST}(\mathbb{R})$  and  $\mathcal{L}$  is the orbit manifold defined by this left action and the Riemannian structure of  $\mathcal{L}$  is defined through the Riemannian structure of  $\text{ST}(\mathbb{R})$ .

We denote by  $[A]$  the  $O(2)$ -orbit defined by  $A \in \text{ST}(\mathbb{R})$  and we denote by  $\text{Span}(A) \subseteq \mathbb{R}^n$  the vector subspace of dimension 2 spanned by the rows of  $A$ .

**Lemma V.15.** *Let  $\pi: \text{ST}(\mathbb{R}) \rightarrow \mathcal{L}$  be the canonical projection onto the orbit space. Then, for every  $A \in \text{ST}(\mathbb{R})$ , the tangent mapping  $T_A \pi: T_A \text{ST}(\mathbb{R}) \rightarrow T_{[A]} \mathcal{L}$  is given by the following identity:*

$$T_A \pi(B) = B(\text{Id}_{n+1} - A^T A).$$

**Proof.** Note that the tangent space to the orbit  $T_A [A] \subseteq T_A \text{ST}(\mathbb{R})$  is identified with the vector space of antisymmetric matrices  $T_{\text{Id}_2} O(2)$  by the isomorphism  $\psi: T_{\text{Id}_2} O(2) \rightarrow T_A [A]$ , given by  $\psi(N) = NA$ . Note that for every  $A \in \text{ST}(\mathbb{R})$  the inverse mapping  $\psi^{-1}$  is given by  $\psi^{-1}(B) = BA^T$ .

As  $T_{[A]} \mathcal{L} \simeq T_A \text{ST}(\mathbb{R}) / T_A [A]$ , the orthogonal complement of  $T_A [A]$  in  $T_A \text{ST}(\mathbb{R})$  can be isometrically identified with  $T_{[A]} \mathcal{L}$ . Thus, the mapping  $T_A \pi: T_A \text{ST}(\mathbb{R}) \rightarrow T_{[A]} \mathcal{L}$  can be isometrically identified with the orthogonal projection onto the orthogonal complement of  $T_A [A]$  in  $T_A \text{ST}(\mathbb{R})$ . Now, for every matrix  $B \in T_A \text{ST}(\mathbb{R})$ , the following decomposition holds:  $T_A \text{ST}(\mathbb{R}) = T_A [A] \oplus^\perp T_A [A]^\perp$ :

$$B = BA^T A + B(\text{Id}_{n+1} - A^T A).$$

This orthogonal projection then satisfies  $T_A \pi(B) = B(\text{Id}_{n+1} - A^T A)$ , as claimed.  $\square$

Then, we conclude:

**Proposition V.16.** *The Grassmannian  $\mathcal{L}$  is a Riemannian manifold whose dimension satisfies:*

$$\dim \mathcal{L} = \dim \text{ST}(\mathbb{R}) - \dim O(2) = 2(n - 1).$$

Moreover, for every  $L = [A] \in \mathcal{L}$ , the tangent space  $T_L \mathcal{L}$  is given by the following equality:

$$T_L \mathcal{L} \simeq \{B \in \mathcal{M}_{2,n+1}(\mathbb{R}), AB^T = BA^T = 0\},$$

where the metric is the one induced by Frobenius metric in  $T_A \text{ST}(\mathbb{R})$ . That is,

$$\langle B_1, B_2 \rangle_F = \text{Tr}(B_1 B_2^T).$$

**Proof.** Since  $AA^T = \text{Id}_2$ , for every  $B \in T_A \text{ST}_{2,n+1}$ , the following equalities hold:

$$B(\text{Id}_{n+1} - A^T A)A^T = B(A^T - A^T) = 0.$$

Analogously, we have  $A(\text{Id}_{n+1} - A^T A)B^T = 0$ . This proves that the image of  $T_A \pi$  is contained in  $\{B \in \mathcal{M}_{2,n+1}(\mathbb{R}), BA^T = AB^T = 0\}$ . Comparing their dimensions we conclude that they are the same vector subspace of dimension  $2(n+1)$ .  $\square$

We now consider the *incidence manifold*  $\mathcal{I}(\mathbb{R})$  given by the following equality:

$$\mathcal{I}(\mathbb{R}) = \{([A], f) \in \mathcal{L} \times S^n, f \in \text{Span}(A)\}.$$

The following are also well-known facts:

**Proposition V.17.** *The incidence manifold  $\mathcal{I}(\mathbb{R})$  is a compact Riemannian manifold whose dimension satisfies:*

$$\dim \mathcal{I}(\mathbb{R}) = \dim \mathcal{L} + 1 = 2n - 1.$$

For every  $([A], f) \in \mathcal{I}(\mathbb{R})$ , the tangent space  $T_{([A], f)} \mathcal{I}(\mathbb{R})$  is given by the following equality:

$$T_{([A], f)} \mathcal{I}(\mathbb{R}) = \{(B, \eta) \in T_{[A]} \mathcal{L} \times T_f S^n, (\text{Id}_n - A^T A)\eta^T = B^T A f^T\},$$

and the metric structure in  $T_{([A], f)} \mathcal{I}(\mathbb{R})$  is the one induced by those of  $T_{[A]} \mathcal{L}$  and  $T_f S^n$ .

**Proof. (Sketch)** Let  $(A(t), f(t))$  be a lifting to  $\text{ST}(\mathbb{R}) \times S^n$  of a smooth curve inside  $\mathcal{I}(\mathbb{R})$ , such that  $(A(0), f(0)) = (A, f)$ . The fact that  $f(t)$  belongs to the vector subspace  $V_t$  spanned by rows of  $A(t)$  may also be written as the fact that the orthogonal projection of  $f(t)$  onto  $V_t$  equals  $f(t)$ . This yields the equation:

$$f^T(t) := A^T(t)A(t)f^T(t).$$

Differentiating at  $t=0$ , we obtain:

$$\dot{f}^T = \dot{A}^T A f^T + A^T \dot{A} f^T + A^T A \dot{f}^T.$$

Thus,  $(B, \eta) \in T_{[A]} \mathcal{L} \times T_f S^n$  is in  $T_{([A], f)} \mathcal{I}(\mathbb{R})$  if, and only if, the following equality is satisfied:

$$(\text{Id}_{n+1} - A^T A)\eta^T = (B^T A + A^T B)f^T.$$

Now, as  $f$  is in the vector space  $V_0$  spanned by the rows of  $A$  and, as  $B \in T_{[A]} \mathcal{L}$ , we conclude that  $AB^T = BA^T = 0$ . We thus conclude that the rows of  $B$  are orthogonal to any vector in  $V_0$  and, in particular, to  $f$ . This yields  $Bf^T = 0$  and, hence,  $A^T Bf^T = 0$ .  $\square$

Let  $\pi_1, \pi_2$  be the restrictions to  $\mathcal{I}(\mathbb{R})$  of the two canonical projections from  $\mathcal{L} \times S^n$ . Namely, we consider the mappings:

$$\pi_1: \mathcal{I}(\mathbb{R}) \longrightarrow \mathcal{L}, \quad \pi_2: \mathcal{I}(\mathbb{R}) \longrightarrow S^n,$$

given by

$$\pi_1([A], f) = [A], \quad \pi_2([A], f) = f.$$

**Proposition V.18.** *With these notation,  $\pi_1$  and  $\pi_2$  are submersions. In particular, for every  $p < n$ , the inverse image  $\mathcal{I}(S^p) = \pi_2^{-1}(S^p)$  is a Riemannian submanifold of  $\mathcal{I}(\mathbb{R})$  whose dimension satisfies:*

$$\dim \mathcal{I}(S^p) = n + p - 1.$$

Moreover, for every  $([A], f) \in \mathcal{I}(S^p)$ , the tangent space  $T_{([A], f)}(\mathcal{I}(S^p))$  satisfies:

$$T_{([A], f)}(\mathcal{I}(S^p)) = T_{([A], f)}\pi_2^{-1}(T_f S^p).$$

Namely, the following equality holds:

$$T_{([A], f)}\mathcal{I}(S^p) = \{(B, \eta) \in T_{[A]}\mathcal{L} \times T_f S^p, (\text{Id}_{n+1} - A^T A)\eta^T = B^T A f^T\},$$

and the Riemannian metric is the one induced as subspace of  $T_{[A]}\mathcal{L} \times T_f S^p$ .

**Proof.** It follows from standard arguments from the fact that  $\pi_2$  is a submersion. The reader may follow them in [Dem89, Chapter III], for instance.  $\square$

## 2.2 The Schubert variety $\mathcal{L}_M$ : Proof of Lemma V.14

**Definition V.19.** *We define the Schubert variety  $\mathcal{L}_M$  as*

$$\mathcal{L}_M = \pi_1(\mathcal{I}(S^p)) = \pi_1(\pi_2^{-1}(S^p)),$$

where we have identified  $S(M)$  as a submanifold of  $S^n$ . Namely,  $\mathcal{L}_M$  is the semialgebraic set of all great circles in  $S^n$  that intersect  $S(M)$ .

Without loss of generality we may assume  $M = \mathbb{R}^{p+1}$ , where  $\mathbb{R}^{p+1}$  is identified with the vector subspace of  $\mathbb{R}^{n+1}$  whose last  $n - p$  coordinates are zero. Accordingly,  $S(M)$  is identified with  $S^p$ .

Let us also define the mapping  $\pi_1^{(2)}: \mathcal{I}(S^p) \longrightarrow \mathcal{L}$  as the restriction

$$\pi_1^{(2)} = \pi_1|_{\pi_2^{-1}(S^p)}.$$

Let  $\mathfrak{C}_M$  be the set of points  $[A] \in \mathcal{L}_M$  such that  $\sharp(\text{Span}(A) \cap S^p) = 2$ . In other words,  $\mathfrak{C}_M$  is the set of great circles in  $S^n$  such that their intersection with  $S^p$  consists of exactly two points  $\pm f$ . Note that  $\mathcal{L}_M \setminus \mathfrak{C}_M$  is the set of great circles in  $S^n$  which are completely embedded in  $S^p$ . In particular,  $\mathcal{L}_M \setminus \mathfrak{C}_M = G_{2, p+1}(\mathbb{R})$  is the Grassmannian of great circles in  $S^p$ . The following proposition implies Lemma V.14.

**Proposition V.20.** *With these notation, the following properties hold:*

1. *For every  $([A], f) \in \mathcal{I}(S^p)$ , the tangent mapping  $T_{([A], f)}\pi_1^{(2)}$  is injective if, and only if,  $[A] \in \mathfrak{C}_M$ . In particular,  $\pi_1^{(2)}: \mathcal{I}(S^p) \longrightarrow \mathcal{L}_M$  is an immersion at every  $([A], f) \in \mathcal{I}(S^p)$  such that  $[A] \in \mathfrak{C}_M$ ;*

2. For every  $[A] \in \mathfrak{C}_M$  and  $([A], f) \in \mathcal{I}(S^p)$  the following properties hold:

- The point  $[A]$  is a regular point of maximal dimension in  $\mathcal{L}_M$ ;
- The mapping  $\pi_1^{(2)}: \mathcal{I}(S^p) \rightarrow \mathcal{L}_M$  is a 2-fold smooth covering map and a submersion in a neighborhood of  $([A], f)$ ;
- The following equality holds:

$$\dim_{([A], f)} \mathcal{I}(S^p) = \dim_{[A]} \mathcal{L}_M = \dim \mathcal{L}_M = n + p - 1.$$

In particular, for every  $[A] \in \mathfrak{C}_M$  the tangent spaces satisfy:

$$T_{[A]} \mathcal{L}_M = T_{([A], f)} \pi_1^{(2)}(T_{([A], f)} \mathcal{I}(S^p)).$$

Namely,

$$T_{[A]} \mathcal{L}_M = \{B \in T_{[A]} \mathcal{L}, \exists \eta \in T_f S^p, (\text{Id}_{n+1} - A^T A) \eta^T = B^T A f^T\},$$

where  $\text{Span}(A) \cap S^p = \{\pm f\}$ .

**Proof.** First of all the following inequalities obviously hold.

$$\dim_{[A]} \mathcal{L}_M \leq \dim \mathcal{L}_M \leq \dim_{([A], f)} \mathcal{I}(S^p) = n + p - 1.$$

There is a natural isometric action of the orthogonal group  $O(n+1)$  on the compact Stiefel manifold  $\text{ST}(\mathbb{R})$  which may be translated to the Grassmannian  $\mathcal{L}$  and, then, to the incidence variety  $\mathcal{I}(\mathbb{R})$  as follows:

$$\begin{aligned} O(n+1) \times \mathcal{I}(\mathbb{R}) &\longrightarrow \mathcal{I}(\mathbb{R}) \\ (U, ([A], f)) &\longmapsto ([AU], fU). \end{aligned}$$

Let us now consider the Lie subgroup  $\mathfrak{D}(p+1, n-p) = O(p+1) \times O(n-p)$  of  $O(n+1)$ . This group acts isometrically both on  $\mathcal{I}(S^p)$  and  $\mathcal{L}_M$ . Up to some isometry defined by some orthogonal matrix  $U \in \mathfrak{D}(p+1, n-p)$ , we may assume

$$([A], f) = \left( \left[ \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & r & 0 & \cdots & 0 & s \end{pmatrix} \right], (1, 0, 0, \dots, 0, 0) \right),$$

where  $r^2 + s^2 = 1$ , and  $s \neq 0$  if, and only if,  $[A] \in \mathfrak{C}_M$ .

Now we prove that  $T_{([A], f)} \pi_1^{(2)}$  is a monomorphism if and only if  $s \neq 0$ . Note that for every  $(B, \eta) \in T_{([A], f)}(\mathcal{I}(S^p))$  the following properties hold:

$$BA^T = 0, \langle \eta, f \rangle = 0, \eta = (x_1, \dots, x_{p+1}, 0, \dots, 0) \in T_f S^n,$$

and

$$(\text{Id}_{n+1} - A^T A) \eta^T = B^T A f^T.$$

Let  $(B, \eta) \in T_{([A], f)}(\mathcal{I}(S^p))$  be in the kernel of  $T_{([A], f)} \pi_1^{(2)}$ . Then,

$$T_{([A], f)} \pi_1^{(2)}(B, \eta) = B = 0$$

and we have:

$$\eta = (0, x_2, \dots, x_{p+1}, 0, \dots, 0), (\text{Id}_{n+1} - A^T A) \eta^T = 0.$$

As  $s^2 + r^2 = 1$ , we also have

$$(\text{Id}_{n+1} - A^T A) = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ 0 & s^2 & \cdots & -r s \\ \vdots & \vdots & \text{Id}_{n-2} & \vdots \\ 0 & -r s & \cdots & r^2 \end{pmatrix}.$$

Hence,

$$0 = (\text{Id}_{n+1} - A^T A) \begin{pmatrix} 0 \\ x_2 \\ x_3 \\ \vdots \\ x_{p+1} \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ s^2 x_2 \\ x_3 \\ \vdots \\ x_{p+1} \\ 0 \\ \vdots \\ 0 \\ -r s x_2 \end{pmatrix}.$$

Thus, if  $s \neq 0$ , we conclude  $\eta = 0$  and  $T_{([A], f)} \pi_1^{(2)}$  is a linear monomorphism. Otherwise, if  $s = 0$ ,  $([0], (0, 1, 0, \dots, 0))$  would be a non-zero element in the kernel of  $T_{([A], f)} \pi_1^{(2)}$ . This proves claim 1 of the proposition.

Recall now that the real Grassmannian  $\mathcal{L}$  may be viewed as an affine semialgebraic set (cf. [BCR98], for instance). Then,  $\mathcal{L}_M$  may also be viewed as a semialgebraic subset of the Grassmannian. As  $\pi_1^{(2)}$  is an immersion at  $([A], f)$ , there is some semialgebraic subset  $V$  of  $\mathcal{L}_M$  containing  $[A]$  and such that  $V$  is diffeomorphic to some open neighborhood of  $([A], f)$  in  $\mathcal{I}(S^p)$ . In particular, we have

$$\begin{aligned} n + p - 1 = \dim_{[A]} V &= \dim_{[A]} \mathcal{I}(S^p) \leq \dim_{[A]} \mathcal{L}_M \\ &\leq \dim \mathcal{L}_M \leq n + p - 1, \end{aligned}$$

for all  $[A] \in \mathfrak{C}_M$  and the last statement of claim 2 holds.

Moreover, for every  $[A] \in \mathfrak{C}_M$  and for every  $f$  such that  $([A], f) \in \mathcal{I}(S^p)$ , there is a compact neighborhood of  $([A], f)$  in  $\mathcal{I}(S^p)$  such that the restriction of  $\pi_1^{(2)}$  to its interior is injective and, hence, a proper embedding. In particular,  $[A]$  is a smooth regular point of  $\mathcal{L}_M$  of maximal dimension and  $\pi_1^{(2)}$  is a 2-fold covering map in a neighborhood of  $[A]$ . This proves the other two statements of claim 2. The last claim of the proposition immediately follows from these facts and the previously proved statements.  $\square$

### 3 Some geometric integration tools

In this section we prove the following statements concerning normal Jacobians of certain mappings we define.

With the same notation as in Section 2 above, let  $M \subseteq \mathbb{R}^{n+1}$  be a real vector subspace of dimension  $p + 1$  and codimension  $k = n - p$  and let  $\Phi: S^n \times S(M) \setminus \text{Diag} \rightarrow \mathcal{L}_M$  be the mapping given by:

$$\Phi(g, f) = L_{(g, f)}, \quad \forall (g, f) \in S^n \times S(M) \setminus \text{Diag},$$

where  $\text{Diag} = \{(g, f), g = \pm f\}$  and  $L_{(g,f)}$  is the great circle containing  $g$  and  $f$ . In terms of classes  $[A]$  modulo  $O(2)$  of matrices  $A$  in the Stiefel manifold, the mapping  $\Phi$  is given by the following rule:

$$\Phi(g, f) = \left[ \frac{f}{\frac{\text{GS}_f(g)}{(1 - \langle f, g \rangle^2)^{1/2}}} \right],$$

where  $\text{GS}_f(g) = g - \langle f, g \rangle f$ .

**Proposition V.21.** *With this notation, for every  $g \in S^n \setminus S(M)$  and  $f \in S(M)$ , the normal Jacobian of  $\Phi$  satisfies:*

$$\text{NJ}_{(g,f)} \Phi = \frac{\partial_M(\Phi(g, f))^n}{d_{\mathbb{P}}(g, S(M))^{n-1}},$$

where  $\partial_M(\Phi(g, f)) = \partial_M(L_{(g,f)}) = \max \{d_{\mathbb{P}}(h, S(M)), h \in L_{(g,f)}\}$ .

With the same notation we define the following incidence variety:

$$\mathcal{IC}(M) = \pi_1^{-1}(\pi_1(\pi_2^{-1}(S(M)))) = \pi_1^{-1}(\mathcal{L}_M) = \{([A], g) \in \mathcal{I}(\mathbb{R}), [A] \in \mathcal{L}_M\}.$$

We have two canonical projections:

$$\begin{aligned} p_1 &= \pi_1|_{\mathcal{IC}(M)}: \mathcal{IC}(M) \longrightarrow \mathcal{L}_M \\ \text{and } p_2 &= \pi_2|_{\mathcal{IC}(M)}: \mathcal{IC}(M) \longrightarrow S^n. \end{aligned}$$

Observe that  $p_1$  is onto and that  $\dim p_1^{-1}(L) = 1$ . Thus,

$$\dim \mathcal{IC}(M) = n + p - 1 + 1 = n + p.$$

The following property holds:

**Proposition V.22.** *With the same notation as above, given  $([A], g) \in \mathcal{IC}(M)$ , such that  $g \in S^n \setminus S(M)$ . Then  $([A], g)$  is a smooth regular point in  $\mathcal{IC}(M)$ ,  $p_1$  and  $p_2$  are submersions at  $([A], g)$  and, if  $\text{Span}(A) \cap S(M) = \{\pm f\}$ , the quotient of the normal Jacobians of  $p_1$  and of  $p_2$  satisfies the following equality:*

$$\frac{\text{NJ}_{([A],g)} p_1}{\text{NJ}_{([A],g)} p_2} = \left( \frac{1}{\|g - \langle f, g \rangle f\|} \right)^{k-1} = \left( \frac{\partial_M([A])}{d_{\mathbb{P}}(g, S(M))} \right)^{k-1},$$

where  $k$  is the codimension of  $M$  in  $\mathcal{R}^{n+1}$ .

With the same notation, for every  $g \in S^n$ , we denote by  $\mathcal{IC}(M)_g$  the fiber by projection  $p_2$  over  $g$ . Namely,  $\mathcal{IC}(M)_g = p_2^{-1}(\{g\})$ . We also prove the following statement.

**Proposition V.23.** *With the same notation, let  $I(g)$  be the following quantity:*

$$I(g) = \int_{(L,g) \in \mathcal{IC}(M)_g} \frac{1}{\partial_M(L)} \frac{\text{NJ}_{(L,g)} p_1}{\text{NJ}_{(L,g)} p_2} d\mathcal{IC}(M)_g.$$

Following the values of the codimension  $k = n - p$ , we have

1. If  $k = 1$ :

$$I(g) = 2 \nu_{p-1} \int_0^1 \frac{(1-t^2)^{\frac{p}{2}-1}}{(1-r^2(1-t^2))^{1/2}} dt,$$

where  $r^2 = 1 - d_{\mathbb{P}}(g, S^p)^2$ . In particular, we have

$$\nu_{p-1} B\left(\frac{1}{2}, \frac{p}{2}\right) \leq I(g) \leq \frac{\nu_{p-1} B\left(\frac{1}{2}, \frac{p}{2}\right)}{d_{\mathbb{P}}(g, S^p)};$$

2. If  $k \in 2\mathbb{N}^*$ , then

$$I(g) = \sum_{i=0}^{\frac{k}{2}-1} \frac{2 \nu_{p-1}}{d_{\mathbb{P}}(g, S^p)^{2i+1}} \frac{B\left(i + \frac{1}{2}, \frac{n}{2} - i - 1\right)}{k B\left(\frac{k}{2} - i, i + 1\right)};$$

3. If  $k \in (2\mathbb{N}^* + 1)$ , then

$$I(g) = \sum_{i=0}^{\infty} \frac{2 \nu_{p-1}}{d_{\mathbb{P}}(g, S^p)^{2i+1}} \frac{B\left(i + \frac{1}{2}, \frac{n}{2} - i - 1\right)}{k B\left(\frac{k}{2} - i, i + 1\right)}.$$

In the latter case, we may also exhibit the following upper and lower bounds given by finite sums:

$$I(g) \leq \sum_{i=0}^{\frac{k-3}{2}} \frac{4 \nu_{p-1}}{d_{\mathbb{P}}(g, S^p)^{2i+2}} \frac{B\left(i + \frac{1}{2}, \frac{n}{2} - i - \frac{3}{2}\right)}{(k-1) B\left(\frac{k-1}{2} - i, i + 1\right)},$$

and

$$I(g) \geq \sum_{i=0}^{\frac{k-3}{2}} \frac{4 \nu_{p-1}}{d_{\mathbb{P}}(g, S^p)^{2i+2}} \frac{B\left(i + 1, \frac{n}{2} - i - \frac{3}{2}\right)}{(k-1) B\left(\frac{k-1}{2} - i, i + 1\right)}.$$

**Remark V.24.** Let  $s = d_{\mathbb{P}}(g, S(M))$  and  $r$  be such that  $r^2 + s^2 = 1$  and let  $F$  be the following function

$$\begin{aligned} F(r, s) &= \int_0^{1/s} (1+r^2 z^2)^{\frac{n-p-2}{2}} (1-s^2 z^2)^{\frac{p-2}{2}} dz \\ &= \frac{1}{d_{\mathbb{P}}(g, S(M))} F_1\left(\frac{1}{2}, \frac{p+2-n}{2}, \frac{2-p}{2}, \frac{3}{2}; -\cot(d_R(g, S^p)), 1\right), \end{aligned}$$

where  $F_1$  is Appell's hypergeometric function and  $\cot(d_R(g, S(M)))$  is the cotangent of the Riemannian distance of  $g$  to  $S(M)$ . Then, quantity  $I(g)$  can be rewritten

$$I(g) = 2 \nu_{p-1} F(r, s).$$

**Remark V.25.** Whenever the codimension is greater than 2, the following bounds hold:

$$\frac{\nu_{p-1}}{d_{\mathbb{P}}(g, S^p)^{k-1}} B\left(\frac{k-1}{2}, \frac{p}{2}\right) \leq I(g) \leq \frac{2 \nu_{p-1}}{d_{\mathbb{P}}(g, S^p)^{k-1}}.$$

Here we follow the same notation as in Section 2 above. In Section 3.3 we prove Proposition V.21, in Section 3.4 we prove Proposition V.22 and in Section 3.5 we prove Proposition V.23.

We assume  $M = \mathbb{R}^{p+1}$  as real vector subspace of  $\mathbb{R}^{n+1}$ ,  $S^p$  is the sphere  $S(M)$  as Riemannian submanifold of  $S^n$ . We denote by  $\mathcal{L}$  the Grassmannian of great circles in  $S^n$  and by  $\mathcal{L}_M$  the semialgebraic subset of  $\mathcal{L}$  given as the lines  $L \in \mathcal{L}$  that intersect  $S(M)$ . Finally,  $\mathfrak{C}_M$  is the manifold given as the subset of  $\mathcal{L}_M$  such that  $\sharp(L \cap S(M)) = 2$ . Before getting into the proofs of these two propositions, we need to establish some basic facts.

### 3.1 Normal Jacobians and the Co-area formula

Our first statement is a classical formula discovered by Federer that can be found in many places in the literature. Some classic references are [Fed69, Mor09, San04]. Our formulation below has been taken from [BCSS98, p. 241].

Let  $X$  and  $Y$  be Riemannian manifolds, and let  $F: X \rightarrow Y$  be a  $C^1$  surjective map. Let  $p = \dim Y$  be the real dimension of  $Y$ . For every point  $x \in X$  such that the tangent mapping  $T_x F$  is surjective, let  $(v_1^x, \dots, v_p^x)$  be an orthonormal basis of  $\ker(T_x F)^\perp$ . Then, we define the normal Jacobian of  $F$  at  $x$ ,  $\text{NJ}_x F$ , as the volume in  $T_{F(x)} Y$  of the parallelepiped spanned by  $(T_x F(v_1^x), \dots, T_x F(v_p^x))$ . In the case that  $T_x F$  is not surjective, we define  $\text{NJ}_x F$  as 0.

Note that, in particular, normal Jacobians remain equal under the action of Riemannian isometries. Namely, the following statement holds:

**Proposition V.26.** *Let  $X, Y$  be two Riemannian manifolds, and let  $F: X \rightarrow Y$  be a  $C^1$  map. Let  $x_1, x_2 \in X$  be two points. Assume that there exist isometries  $\varphi_X: X \rightarrow X$  and  $\varphi_Y: Y \rightarrow Y$  such that  $\varphi_X(x_1) = x_2$ , and*

$$F \circ \varphi_X = \varphi_Y \circ F.$$

*Then, the following equality holds:*

$$\text{NJ}_{x_1} F = \text{NJ}_{x_2} F.$$

*Moreover, if there exists an inverse  $G: Y \rightarrow X$ , then*

$$\text{NJ}_x F = \frac{1}{\text{NJ}_{F(x)} G}.$$

**Theorem V.27. (Co-area formula)** *Consider a differentiable map  $F: X \rightarrow Y$ , where  $X$  and  $Y$  are Riemannian manifolds of respective real dimensions  $n \geq p$ . Consider a measurable function  $f: X \rightarrow \mathbb{R}$ , such that  $f$  is integrable. Then, for every  $y \in Y$  except in a zero-measure set,  $F^{-1}(y)$  is empty or a real submanifold of  $X$  of real dimension  $n - p$ . Moreover, the following equality holds (and the integrals appearing on it are well-defined):*

$$\int_X f \text{NJ}_x F \, dX = \int_{y \in Y} \left( \int_{x \in F^{-1}(y)} f(x) \, dF^{-1}(y) \right) dY,$$

*where  $\text{NJ}_x F$  is the normal Jacobian of  $F$  in  $x$ .*

### 3.2 Distances in $\mathcal{L}_M$ : Some technical results

We denote by  $d_{\mathbb{P}}: (S^n)^2 \rightarrow \mathbb{R}_+$  the “projective” distance on the sphere as in [BCSS98] (i.e.  $d_{\mathbb{P}}(f, g) = \sin d_R(f, g)$ , where  $d_R(f, g)$  is the standard Riemannian (arclength) distance in  $S^n$ ).

Let  $L = [A] \in \mathfrak{C}_M$  be a great circle that intersects  $S^p$  in exactly two points. Assume  $\text{Span}(A) \cap S^p = \{\pm f\}$ . Up to some isometry in  $O(p+1) \times O(n-p)$  we may assume that  $f = (1, 0, \dots, 0)$  and that

$$L = [A] = \left[ \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & r & 0 & \cdots & 0 & s \end{pmatrix} \right],$$

where  $r^2 + s^2 = 1$ . Moreover, the following mapping is an isometry between  $L$  and  $S^1$ :

$$\begin{aligned} \varphi: S^1 &\longrightarrow L, \\ (\lambda, \mu) &\longmapsto (\lambda, \mu r, 0, \dots, 0, \mu s). \end{aligned}$$

**Lemma V.28.** *With this notation, let  $g = \varphi(\lambda, \mu)$  be any point in  $L$ , then the following properties hold:*

- $d_{\mathbb{P}}(g, S^p) = |\mu s|$ ,
- $\partial_M(L) = \max \{d_{\mathbb{P}}(g, S^p), g \in L\} = |s|$ ,
- $\frac{d_{\mathbb{P}}(g, S^p)}{\partial_M(L)} = |\mu| = \|g - \langle f, g \rangle f\| = (1 - \langle f, g \rangle^2)^{1/2}$ .

The proof comes from simple calculations. The following statement also holds:

**Lemma V.29.** *For every  $L \in \mathfrak{C}_M$ , the following equality holds for every positive integer  $r \in \mathbb{N}$ ,  $r \geq 2$ :*

$$I_r(L) = \int_L d_{\mathbb{P}}(x, S^p)^r dL = \frac{\nu_{r+2}}{\nu_{r+1}} \partial_M(L)^r = B\left(\frac{r+3}{2}, \frac{1}{2}\right) \partial_M(L)^r,$$

where  $\nu_r$  is the volume of the  $r$ th dimensional sphere, namely

$$\nu_r = \text{vol}[S^r] = \frac{\pi^{r/2}}{\Gamma(\frac{r}{2} + 1)}.$$

**Proof.** Using the isometry  $\varphi$  above, we have  $d_{\mathbb{P}}(g, S^p) = |s \mu| = \partial_M(L) |\mu|$  and hence, we have:

$$I_r(L) = \partial_M(L)^r \int_{S^1} |\mu|^r d\nu_{S^1}.$$

Now, we project  $\pi: S^1 \rightarrow [-1, 1]$ , where  $\pi(\lambda, \mu) = \mu$ . The normal Jacobian  $\text{NJ}_x \pi$  equals  $(1 - |\pi(x)|^2)^{1/2}$  (cf. [BCSS98, p. 206], for instance) and we use the Co-area formula to conclude:

$$I_r(L) = \partial_M(L)^r \int_{-1}^1 \frac{|\mu|^r}{(1 - \mu^2)^{1/2}} d\mu = 2 \partial_M(L)^r \int_0^1 \frac{\mu^r}{(1 - \mu^2)^{1/2}} d\mu.$$

The following equality is classical (*cf.* [Cho00], for instance) and finishes the proof:

$$2 \int_0^1 \frac{\mu^r}{(1-\mu^2)^{1/2}} d\mu = \frac{\nu_{r+2}}{\nu_{r+1}}. \quad \square$$

We may define a density function on every great circle  $L \in \mathfrak{C}_M$ . We denote  $dL^{(M)}$  the probability distribution defined in the following terms. For every integrable function  $\Phi: S^n \rightarrow \mathbb{R}_+$ , we define:

$$E_{L^{(M)}}[\Phi] = \int_L \Phi dL^{(M)} = \frac{\nu_k}{\nu_{k+1} \partial_M(L)} \int_L \Phi(x) d_{\mathbb{P}}(x, S^p)^{k-1} dL,$$

where  $k = n - p$  is the codimension of  $M$  in  $\mathbb{R}^{n+1}$ .

### 3.3 Normal Jacobians I: Proof of Proposition V.21

We follow the same notation as in previous sections and subsections.

As the normal Jacobian is invariant under the action of isometries (Proposition V.26 above), we may assume that

$$f = (1, 0, \dots, 0) \in S^p, g = (\lambda, \mu r, 0, \dots, 0, \mu s) \in S^n,$$

where  $r^2 + s^2 = 1$  and  $\lambda^2 + \mu^2 = 1$ . Hence,

$$\Phi(g, f) = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & r & 0 & \cdots & 0 & s \end{bmatrix}.$$

We may decompose  $\Phi = \pi \circ \varphi$  as the composition of the following two mappings:

- A first mapping into the Stiefel manifold:

$$\begin{aligned} \varphi: S^n \times S^p \setminus \text{Diag} &\longrightarrow \text{ST}(\mathbb{R}) \\ (h_1, h_2) &\longmapsto \begin{pmatrix} h_2 \\ \frac{\text{GS}_{h_2}(h_1)}{(1 - \langle h_1, h_2 \rangle)^{1/2}} \end{pmatrix}, \end{aligned}$$

where  $\text{GS}_{h_2}(h_1) = h_1 - \langle h_1, h_2 \rangle h_2$  was defined above.

- The canonical projection  $\pi: \text{ST}(\mathbb{R}) \rightarrow \text{ST}(\mathbb{R})/O(2) = \mathcal{L}$ . In this case the tangent mapping  $T_A \pi$  is the orthogonal projection of Lemma V.15 above, and it is given by the following matrix:

$$\text{Id}_{n+1} - A^T A = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ 0 & s^2 & \cdots & -r s \\ \vdots & \vdots & \text{Id}_{n-2} & \vdots \\ 0 & -r s & \cdots & r^2 \end{pmatrix}.$$

Then, for every  $(\dot{g}, \dot{f}) \in T_g S^n \times T_f S^p$ , the following equality holds:

$$T_{(g,f)} \Phi(\dot{g}, \dot{f}) = T_A \pi(T_{(g,f)} \varphi(\dot{g}, \dot{f})),$$

where  $A = \varphi(g, f)$ .

We start by computing the tangent mapping  $T_{(g,f)} \varphi$ , which is given by the following identities:

$$T_{(g,f)} \varphi: T_g S^n \times T_f S^p \longrightarrow T_{\varphi(g,f)} \text{ST}(\mathbb{R})$$

$$T_{(g,f)} \varphi(\dot{g}, \dot{f}) \longmapsto \left( \begin{array}{c} \dot{f} \\ \left( \frac{\text{GS}_{\dot{f}}(g)}{(1 - \langle f, g \rangle^2)^{1/2}} \right) \end{array} \right),$$

where

$$\left( \frac{\text{GS}_{\dot{f}}(g)}{(1 - \langle f, g \rangle^2)^{1/2}} \right) = \frac{(1 - \langle f, g \rangle^2)^{1/2} \rho_{\dot{f}, \dot{g}}(f, g) + (1 - \langle f, g \rangle^2)^{-1/2} \tau_{\dot{f}, \dot{g}}(f, g)}{(1 - \langle f, g \rangle^2)},$$

and

$$\rho_{\dot{f}, \dot{g}}(f, g) = \text{GS}_f(\dot{g}) - (\langle g, \dot{f} \rangle f + \langle g, f \rangle \dot{f}) = \dot{g} - \langle \dot{g}, f \rangle f - (\langle g, \dot{f} \rangle f + \langle g, f \rangle \dot{f}),$$

$$\tau_{\dot{f}, \dot{g}}(f, g) = \langle f, g \rangle [\langle g, \dot{f} \rangle + \langle \dot{g}, f \rangle] \text{GS}_f(g).$$

Now we consider the following orthonormal bases of the tangent spaces  $T_f S^p$  and  $T_g S^n$ :

- $T_f S^p$  is generated by the list of tangent vectors  $\{\dot{f}_2, \dots, \dot{f}_{p+1}\}$  where  $\dot{f}_i$  is the vector whose coordinates are all zero excepting the  $i$ th coordinate which is 1. Therefore  $f = f_1$ .
- $T_g S^n$  is generated by the list of tangent vectors  $\{\dot{g}_1, \dots, \dot{g}_n\}$ , where
  - $\dot{g}_1 = (-\mu, \lambda r, 0, \dots, 0, \lambda s)$ ,
  - $\dot{g}_2 = (0, s, 0, \dots, 0, -r)$ ,
  - and for every  $i$ ,  $3 \leq i \leq n$ ,  $\dot{g}_i$  is the vector whose coordinates are all zero excepting the  $i$ th coordinate which is 1.

Now some calculations would yield

- For every  $i$ ,  $3 \leq i \leq p+1$ , we have

$$T_{(g,f)} \varphi(0, \dot{f}_i) = \left( \begin{array}{c} \dot{f}_i \\ -\frac{\langle f, g \rangle}{(1 - \langle f, g \rangle^2)^{1/2}} \dot{f}_i \end{array} \right) = \left( \begin{array}{c} \dot{f}_i \\ -\frac{\lambda}{\mu} \dot{f}_i \end{array} \right).$$

- As for the case  $i = 2$  we have:

$$T_{(g,f)} \varphi(0, \dot{f}_i) = \left( \begin{array}{c} \dot{f}_i \\ u_1 \end{array} \right),$$

where

$$u_1 = \left( -r, -\frac{\lambda}{\mu} s^2, 0, \dots, 0, \frac{\lambda}{\mu} r s \right).$$

- For every  $j$ ,  $3 \leq j \leq n$ , we have

$$T_{(g,f)} \varphi(\dot{g}_j, 0) = \left( \begin{array}{c} 0 \\ \frac{1}{\mu} \dot{g}_j \end{array} \right).$$

- For  $j = 1$  we have

$$T_{(g,f)} \varphi(\dot{g}_1, 0) = \begin{pmatrix} 0 \\ u_2 \end{pmatrix},$$

where

$$u_2 = \lambda \mu (0, r, 0, \dots, 0, s).$$

- Finally, for  $j = 2$ , we have

$$T_{(g,f)} \varphi(\dot{g}_2, 0) = \begin{pmatrix} 0 \\ \frac{1}{\mu} \dot{g}_3 \end{pmatrix}.$$

Now we consider the following matrices in  $T_{\varphi(g,f)} \text{ST}(\mathbb{R})$  which are part of an orthonormal basis with respect to Frobenius inner product. In fact, all of them belong to  $T_{\Phi(g,f)} \mathfrak{C}_M$  and also to  $T_{\Phi(g,f)} L$ .

- The matrix  $E_{1,2}$  given by:

$$E_{1,2} = \begin{pmatrix} 0 & s & 0 & \cdots & 0 & -r \\ 0 & 0 & 0 & \cdots & 0 & 0 \end{pmatrix}.$$

- For every  $i$ ,  $3 \leq i \leq p+1$ , let  $E_{1,i}$  be the matrix given as:

$$E_{1,i} = \begin{pmatrix} \dot{f}_i \\ 0 \end{pmatrix}.$$

- The matrix  $E_{2,2}$  given by

$$E_{2,2} = \begin{pmatrix} 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & s & 0 & \cdots & 0 & -r \end{pmatrix}.$$

- For every  $j$ ,  $3 \leq j \leq n$ , let  $E_{2,j}$  be the matrix given as:

$$E_{2,j} = \begin{pmatrix} 0 \\ \dot{g}_j \end{pmatrix}.$$

Now, we have:

- For every  $i$ ,  $3 \leq i \leq p+1$ ,

$$T_{(g,f)} \Phi(0, \dot{f}_i) = T_A \pi(T_{(g,f)} \varphi(0, \dot{f}_i)) = T_{(g,f)} \varphi(0, \dot{f}_i) = E_{1,i} - \frac{\lambda}{\mu} E_{2,i}.$$

- For  $i = 2$ ,

$$\begin{aligned} T_{(g,f)} \Phi(0, \dot{f}_2) &= T_A \pi(T_{(g,f)} \varphi(0, \dot{f}_2)) = T_{(g,f)} \varphi(0, \dot{f}_2) (\text{Id}_{n+1} - A^T A) \\ &= s E_{1,2} + \frac{\lambda s}{\mu} E_{2,2}. \end{aligned}$$

- For every  $j$ ,  $3 \leq j \leq n$ ,

$$T_{(g,f)} \Phi(\dot{g}_j, 0) = T_A \pi(T_{(g,f)} \varphi(\dot{g}_j, 0)) = T_{(g,f)} \varphi(\dot{g}_j, 0) = \frac{1}{\mu} E_{2,j}.$$

- For  $j = 1$ ,

$$T_{(g,f)} \Phi(\dot{g}_2, 0) = T_A \pi(T_{(g,f)} \varphi(\dot{g}_2, 0)) = T_{(g,f)} \varphi(\dot{g}_2, 0) (\text{Id}_{n+1} - A^T A) = 0.$$

- Finally, for  $j = 2$ ,

$$T_{(g,f)} \Phi(\dot{g}_2, 0) = T_A \pi(T_{(g,f)} \varphi(\dot{g}_2, 0)) = T_{(g,f)} \varphi(\dot{g}_2, 0) = \frac{1}{\mu} E_{2,2}.$$

In particular, we conclude that the kernel of  $T_{(g,f)} \Phi$  is the vector subspace generated by  $(\dot{g}_2, 0) \in T_g S^n \times T_f S^p$ . The restriction of  $T_{(g,f)} \Phi$  to the orthogonal complement of its kernel, taking orthonormal basis, is given by a triangular matrix of the following form:

$$\begin{pmatrix} s & * & * \\ 0 & \text{Id}_{p-1} & * \\ 0 & 0 & \frac{1}{\mu} \text{Id}_{n-1} \end{pmatrix}.$$

Then, the normal Jacobian satisfies

$$\text{NJ}_{(g,f)} \Phi = \frac{|s|}{\mu^{n-1}} = \frac{\partial_M(L)^n}{d_{\mathbb{P}}(g, S^p)^{n-1}},$$

as wanted.  $\square$

### 3.4 Normal Jacobians II: Proof of Proposition V.22

Once again we follow the same notation as above.

First of all, observe that if  $([A], g) \in \mathcal{IC}(M)$ , then  $[A] \in \mathfrak{C}_M$  and this is a smooth point of maximal dimension in  $\mathcal{L}_M$ . Now, we proceed by computing the tangent space  $T_{([A],g)} \mathcal{IC}(M)$ . Again, due to the right action of  $O(p+1) \times O(n-p)$  on  $\mathcal{I}(S^p)$  and  $\mathcal{I}(\mathbb{R})$ . Since Proposition V.26 about the invariance of normal Jacobians holds, we may assume:

$$([A], g) = \left( \left[ \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & r & 0 & \cdots & 0 & s \end{pmatrix} \right], (\lambda, \mu r, 0, \dots, 0, \mu s) \right),$$

where  $r^2 + s^2 = 1$ ,  $\lambda^2 + \mu^2 = 1$ ,  $\mu \neq 0$  (since  $g \notin S^p$ ) and (then)  $s \neq 0$ . Let us also write  $f = (1, 0, 0, \dots, 0) \in S^p \cap \text{Span}(A)$ . Observe that  $\|g - \langle f, g \rangle f\| = (1 - \langle f, g \rangle^2)^{1/2} = |\mu|$ . For sake of simplicity, assume  $\mu \geq 0$  from now on.

We need to compute an orthonormal basis of  $T_{([A],g)} \mathcal{IC}(M)$  and then its images under the two projections  $T_{([A],g)} p_1$  and  $T_{([A],g)} p_2$ . This is done in the following technical lemma:

**Lemma V.30.** *Let  $v_1 = (0, -s, 0, \dots, 0, r)$ ,  $v_2 = (\mu, -\lambda r, 0, \dots, 0, -\lambda s)$  and  $(e_1, \dots, e_{n+1})$  be the canonical orthonormal basis of  $\mathbb{R}^{n+1}$ . Let  $(\omega_1, \dots, \omega_{n+1})$  and  $(\omega'_1, \omega'_3, \dots, \omega'_{p+1})$  be defined as follows:*

- $\omega_1 = \left( \begin{pmatrix} 0 & -s\lambda & 0 & \cdots & 0 & r\lambda \\ 0 & -s\mu & 0 & \cdots & 0 & r\mu \end{pmatrix}, v_1 \right),$
- $\omega_2 = \left( \begin{pmatrix} 0 & \cdots & 0 \\ 0 & \cdots & 0 \end{pmatrix}, v_2 \right),$

- $\omega_i = \left( \begin{pmatrix} 0 & \cdots & 0 & \lambda & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \mu & 0 & \cdots & 0 \end{pmatrix}, e_i \right)$ , for  $3 \leq i \leq p+1$ ,
- $\omega_j = \left( \begin{pmatrix} 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \mu^{-1} & 0 & \cdots & 0 \end{pmatrix}, e_j \right)$ , for  $p+2 \leq j \leq n$ ,
- $\omega'_1 = \left( \begin{pmatrix} 0 & s\mu & 0 & \cdots & 0 & -r\mu \\ 0 & -s\lambda & 0 & \cdots & 0 & r\lambda \end{pmatrix}, 0 \right)$ ,
- $\omega'_i = \left( \begin{pmatrix} 0 & \cdots & 0 & -\mu & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \lambda & 0 & \cdots & 0 \end{pmatrix}, 0 \right)$ , for  $3 \leq i \leq p+1$ .

Then, the following family is an orthonormal basis of  $T_{([A],g)}\mathcal{IC}(M)$ :

$$\beta = \left\{ \frac{1}{\sqrt{2}}\omega_1, \omega_2, \frac{1}{\sqrt{2}}\omega_3, \dots, \frac{1}{\sqrt{2}}\omega_{p+1}, \frac{1}{\sqrt{1+\mu^{-2}}}\omega_{p+2}, \dots, \frac{1}{\sqrt{1+\mu^{-2}}}\omega_n \right\} \\ \cup \{\omega'_1, \omega'_3, \dots, \omega'_{p+1}\}.$$

**Proof.** From Section 2 we have the following description of  $T_{([A],g)}\mathcal{IC}(M)$ :

A pair  $(B, \eta) \in T_{[A]}\mathcal{L} \times T_g S^n$  is in the tangent space  $T_{([A],g)}\mathcal{IC}(M)$  if, and only if, the following properties hold:

1.  $BA^T = 0$ , since  $B \in T_{[A]}\mathcal{L}$ ;
2.  $\langle \eta, g \rangle = 0$ , since  $\eta \in T_g S^n$ ,
3.  $(\text{Id}_{n+1} - A^T A) \eta^T = B^T A g^T$ , since  $(B, \eta) \in T_{([A],g)}\mathcal{I}(\mathbb{R})$ ;
4. There exists  $\nu \in T_f S^p$ , such that  $B = T_{([A],f)}\pi_1^{(2)}(B, \nu)$ . As,  $B$  already satisfies property (1) above, this may be rewritten as:

$$\exists \nu \in T_f S^p, (\text{Id}_{n+1} - A^T A) \nu^T = B^T A g^T.$$

Let us rewrite these properties in terms of matrices and coordinates to prove that  $\beta$  is an orthonormal basis of  $T_{([A],g)}\mathcal{IC}(M)$ .

The condition  $BA^T = 0$  implies that we may assume

$$B = \begin{pmatrix} 0 & -s x_2 & b_{1,3} & \cdots & r x_2 \\ 0 & -s y_2 & b_{2,3} & \cdots & r y_2 \end{pmatrix}.$$

Let  $e_i$ ,  $1 \leq i \leq n+1$  be the canonical (usual) orthonormal basis of  $\mathbb{R}^{n+1}$  and let  $v_1 = (0, -s, \dots, r)$  and  $v_2 = (\mu, -\lambda r, 0, \dots, -\lambda s)$ . The following family is an orthonormal basis of  $T_g S^n$ :

$$\beta = \{v_1, v_2, e_3, \dots, e_n\}.$$

As  $A g^T = \begin{pmatrix} \lambda \\ \mu \end{pmatrix}$ , we conclude

$$B^T A g^T = \begin{pmatrix} 0 \\ (-s)(\lambda x_2 + \mu y_2) \\ \lambda b_{1,3} + \mu b_{2,3} \\ \vdots \\ \lambda b_{1,n} + \mu b_{2,n} \\ (r)(\lambda x_2 + \mu y_2) \end{pmatrix}.$$

Hence, property (3) may be rewritten as:

$$(\text{Id}_{n+1} - A^T A) \eta^T = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ 0 & s^2 & \cdots & -r s \\ \vdots & \vdots & \text{Id}_{n-2} & \vdots \\ 0 & -r s & \cdots & r^2 \end{pmatrix} \eta^T = \begin{pmatrix} 0 \\ (-s) (\lambda x_2 + \mu y_2) \\ \lambda b_{1,3} + \mu b_{2,3} \\ \vdots \\ \lambda b_{1,n} + \mu b_{2,n} \\ (r) (\lambda x_2 + \mu y_2) \end{pmatrix}.$$

Observe that  $(\text{Id}_{n+1} - A^T A) v_1^T = v_1^T$  and  $(\text{Id}_{n+1} - A^T A) v_2^T = 0$ . Hence, assuming that  $\eta = z_1 v_1 + z_2 v_2 + \sum_{i=3}^n z_i e_i$ , property (3) becomes:

$$\begin{pmatrix} 0 \\ (-s) z_1 \\ z_3 \\ \vdots \\ z_n \\ r z_1 \end{pmatrix} = \begin{pmatrix} 0 \\ (-s) (\lambda x_2 + \mu y_2) \\ \lambda b_{1,3} + \mu b_{2,3} \\ \vdots \\ \lambda b_{1,n} + \mu b_{2,n} \\ (r) (\lambda x_2 + \mu y_2) \end{pmatrix}.$$

Now we consider property (4). Since  $\nu \in T_f S^p$ , we may assume that

$$\nu = (0, u_2, \dots, u_{p+1}, 0, \dots, 0) \in \mathbb{R}^{n+1}.$$

As  $A f^T = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ , property (4) may be rewritten as:

$$\begin{pmatrix} 0 & 0 & \cdots & 0 \\ 0 & s^2 & \cdots & -r s \\ \vdots & \vdots & \text{Id}_{n-2} & \vdots \\ 0 & -r s & \cdots & r^2 \end{pmatrix} \begin{pmatrix} 0 \\ u_2 \\ u_3 \\ \vdots \\ u_{p+1} \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ s^2 u_2 \\ u_3 \\ \vdots \\ u_{p+1} \\ 0 \\ \vdots \\ 0 \\ -r s u_2 \end{pmatrix} = \begin{pmatrix} 0 \\ -s x_2 \\ b_{1,3} \\ \vdots \\ b_{1,n} \\ r x_2 \end{pmatrix}.$$

This yields these equalities

$$\begin{aligned} -s u_2 &= x_2, \\ b_{1,j} &= 0, \quad p+2 \leq j \leq n. \end{aligned}$$

Putting all these properties together, we get the following characterization of tangent space  $T_{([A],g)} \mathcal{IC}(M)$ :

$$\left( \begin{pmatrix} 0 & -s x_2 & b_{1,3} & \cdots & r x_2 \\ 0 & -s y_2 & b_{2,3} & \cdots & r y_2 \end{pmatrix}, \eta \right) \in T_{([A],g)} \mathcal{IC}(M)$$

if, and only if, the following properties hold:

- $b_{1,j} = 0, \quad p+2 \leq j \leq n,$
- $\eta = z_1 v_1 + z_2 v_2 + \sum_{i=3}^n z_i e_i,$
- $\lambda x_2 + \mu y_2 = z_1,$

- $\lambda b_{1,i} + \mu b_{2,i} = z_i, 3 \leq i \leq p+1,$
- $\mu b_{2,j} = z_j, p+2 \leq j \leq n.$

The collection of vectors in  $\beta$  described in the statement of the lemma satisfies these properties, they are linearly independent and a family of orthonormal vectors with the accurate number of elements (equal to the dimension of  $T_{([A],g)}\mathcal{IC}(M)$ ) as wanted.  $\square$

Then, note that  $\ker(T_{([A],g)} p_1) = \text{Span}(\{\omega_2\})$  and  $T_{([A],g)} p_1(B, \eta) = B$ . Then, using this orthonormal basis, we immediately compute the list of vectors in  $T_{([A],g)} p_1(\beta)$ . They are mutually orthogonal and we may compute the normal Jacobian as the product of their norms, yielding the following equality:

$$\text{NJ}_{([A],g)} p_1 = \left(\frac{1}{\sqrt{2}}\right)^p \left(\frac{\mu^{-1}}{\sqrt{1+\mu^{-2}}}\right)^{n-p-1} = \left(\frac{1}{\sqrt{2}}\right)^p \left(\frac{1}{\sqrt{1+\mu^2}}\right)^{n-p-1}.$$

On the other hand,

$$\ker(T_{([A],g)} p_2) = \text{Span}(\{\omega'_1, \omega'_3, \dots, \omega'_{p+1}\}), \text{ and } T_{([A],g)} p_2(B, \eta) = \eta.$$

Again, we may compute the list of vectors in  $T_{([A],g)} p_2(\beta)$  and then compute the corresponding normal Jacobian, obtaining:

$$\text{NJ}_{([A],g)} p_2 = \left(\frac{1}{\sqrt{2}}\right)^p \left(\frac{1}{\sqrt{1+\mu^{-2}}}\right)^{n-p-1}.$$

Then, the quotient satisfies:

$$\frac{\text{NJ}_{([A],g)} p_1}{\text{NJ}_{([A],g)} p_2} = \frac{\left(\frac{1}{\sqrt{2}}\right)^p \left(\frac{\mu^{-1}}{\sqrt{1+\mu^{-2}}}\right)^{n-p-1}}{\left(\frac{1}{\sqrt{2}}\right)^p \left(\frac{1}{\sqrt{1+\mu^{-2}}}\right)^{n-p-1}} = \left(\frac{1}{\mu}\right)^{n-p-1},$$

which proves Proposition V.22 as wanted.  $\square$

### 3.5 Fibers over “complex” points: Proof of Proposition V.23

We begin with the following statement.

**Proposition V.31.** *With the same notation as above, for every  $g \in S^n \setminus S^p$ , there is an isometry*

$$\Psi_g: S^p \longrightarrow \mathcal{IC}(M)_g.$$

*In particular, the volume of the fiber  $\mathcal{IC}(M)_g$  is constant and independent of  $g$ . In fact,*

$$\text{vol}[\mathcal{IC}(M)_g] = \nu_p = \text{vol}[S^p].$$

**Proof.** Simply observe that the following mapping is an isometry, an immersion and its image is the fiber  $\mathcal{IC}(M)_g$ , where  $g = (0, r, 0, \dots, 0, s)$ ,  $r^2 + s^2 = 1$ ,  $s \neq 0$ :

$$\Psi_g: S^p \longrightarrow \mathcal{I}(\mathbb{R}),$$

given by

$$\Psi_g(x_1, \dots, x_{p+1}) = \left( \begin{bmatrix} x_1 & -s x_2 & x_3 & \cdots & x_{p+1} & 0 & \cdots & 0 & r x_2 \\ 0 & r & 0 & \cdots & 0 & 0 & \cdots & 0 & s \end{bmatrix}, g \right).$$

First of all, it is clear that  $\Psi_g(x) \in \mathcal{IC}(M)_g$  for all  $x \in S^p$ . The matrix

$$\psi(x) = \begin{pmatrix} x_1 & -s x_2 & x_3 & \cdots & x_{p+1} & 0 & \cdots & 0 & r x_2 \\ 0 & r & 0 & \cdots & 0 & 0 & \cdots & 0 & s \end{pmatrix}$$

is in the Stiefel manifold  $ST(\mathbb{R})$  and so the orbit  $\Psi(x) = [\psi(x)]$  is in the Grassmanian  $\mathcal{L}$ . But observe that

$$\left( (x_1, -s x_2, x_3, \dots, x_{p+1}, 0, \dots, 0, r x_2) - \frac{r x_2}{s} g \right) \in \text{Span}(\psi(x)) \cap \mathbb{R}^{p+1} \neq \emptyset.$$

Thus  $\Psi_g(x) \in \mathcal{IC}(M) \cap p_2^{-1}(g)$  as wanted.

Additionally, observe that the tangent mapping is given by

$$T_x \Psi_g(\eta) = \left( \begin{pmatrix} \eta_1 & -s \eta_2 & \eta_3 & \cdots & \eta_{p+1} & 0 & \cdots & 0 & r \eta_2 \\ 0 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 \end{pmatrix}, 0 \right),$$

where  $\eta = (\eta_1, \dots, \eta_{p+1}) \in x^\perp = T_x S^p$  is orthogonal to  $x$ . Moreover, for  $\eta, \eta' \in T_x S^p$  we have:

$$\langle T_x \Psi_g(\eta), T_x \Psi_g(\eta') \rangle = \eta_1 \eta'_1 + s^2 \eta_2 \eta'_2 + \sum_{i=3}^{p+1} \eta_i \eta'_i + r^2 \eta_2 \eta'_2 = \langle \eta, \eta' \rangle,$$

and  $\Psi_g$  is an isometry. Then, its normal Jacobian is 1 and the equality between the corresponding volumes holds.  $\square$

**Corollary V.32.** *For every point  $g \in S^n \setminus S^p$  and for every couple  $([A], g) \in \mathcal{IC}(M)$ , the quotient of normal Jacobians satisfies*

$$\frac{\text{NJ}_{([A],g)} p_1}{\text{NJ}_{([A],g)} p_2} = \left( \frac{1}{s^{n-p-1}} \right) (s^2 + r^2 x_2^2)^{\frac{n-p-1}{2}},$$

where  $x = (x_1, x_2, x_3, \dots, x_{p+1}) \in S^p$  is such that  $\Psi_g(x) = ([A], g)$ ,  $s^2 = d_{\mathbb{P}}(g, S^p)^2$  and  $r^2 + s^2 = 1$ .

**Proof.** According to Proposition V.22, the quotient of normal Jacobians satisfies:

$$\frac{\text{NJ}_{([A],g)} p_1}{\text{NJ}_{([A],g)} p_2} = \left( \frac{1}{\|g - \langle f, g \rangle f\|} \right)^{n-p-1} = \left( \frac{1}{1 - \langle f, g \rangle^2} \right)^{\frac{n-p-1}{2}},$$

where  $\text{Span}(A) \cap S^p = \{\pm f\}$ . With the same notation as in the proof of the previous proposition, we may assume  $g = (0, r, 0, \dots, 0, s)$ ,  $r^2 + s^2 = 1$ ,  $s \neq 0$ , and  $\Psi_g(x) = ([A], g)$ . Thus, we have seen that

$$v = \left( (x_1, -s x_2, x_3, \dots, x_{p+1}, 0, \dots, 0, r x_2) - \frac{r x_2}{s} g \right) \in \text{Span}(A) \cap \mathbb{R}^{p+1},$$

and, hence we may choose

$$f = \frac{v}{\|v\|},$$

to compute the normal Jacobian. Observe that

$$v = \left( x_1, -\frac{x_2}{s}, x_3, \dots, x_{p+1}, 0, \dots, 0 \right),$$

and

$$\|v\|^2 = 1 + \left( \frac{1}{s^2} - 1 \right) x_2^2 = 1 + \frac{r^2 x_2^2}{s^2},$$

whereas

$$\langle v, g \rangle^2 = \frac{r^2 x_2^2}{s^2}.$$

Hence

$$1 - \langle f, g \rangle^2 = 1 - \frac{\langle v, g \rangle^2}{\|v\|^2} = 1 - \frac{\frac{r^2 x_2^2}{s^2}}{1 + \frac{r^2 x_2^2}{s^2}} = \frac{s^2}{s^2 + r^2 x_2^2}.$$

Finally, we conclude:

$$\frac{N J_{([A], g)} p_1}{N J_{([A], g)} p_2} = \left( \frac{1}{1 - \langle f, g \rangle^2} \right)^{\frac{n-p-1}{2}} = \left( \frac{s^2 + r^2 x_2^2}{s^2} \right)^{\frac{n-p-1}{2}},$$

as wanted.  $\square$

### 3.5.1 Proof of Proposition V.23

As in the proof of Proposition V.31, assuming that  $x = (x_1, x_2, \dots, x_m)$  and  $g = (0, r, 0, \dots, 0, s)$ ,  $r^2 + s^2 = 1$ ,  $s \neq 0$ , we have

$$I(g) = \frac{1}{d_{\mathbb{P}}(g, S^p)} \left( \int_{x \in S^p} \left( \frac{s^2 + r^2 x_2^2}{s^2} \right)^{\frac{n-p-2}{2}} dS^p \right).$$

Integrating in polar coordinates we get:

$$I(g) = \frac{1}{d_{\mathbb{P}}(g, S^p)} \int_{-1}^1 \left( \int_{S^{p-1}_{\sqrt{1-t^2}}} dS^{p-1} \right) \left( \frac{s^2 + r^2 t^2}{s^2} \right)^{\frac{n-p-2}{2}} (1-t^2)^{-1/2} dt.$$

Then,

$$I(g) = \frac{2 \nu_{p-1}}{d_{\mathbb{P}}(g, S^p)} \int_0^1 \left( \frac{s^2 + r^2 t^2}{s^2} \right)^{\frac{n-p-2}{2}} (1-t^2)^{\frac{p-2}{2}} dt. \quad (\text{V.2})$$

In other words,

$$I(g) = \frac{2 \nu_{p-1}}{d_{\mathbb{P}}(g, S^p)} \int_0^1 \left( (1-t^2) + \frac{t^2}{s^2} \right)^{\frac{k}{2}-1} (1-t^2)^{\frac{p}{2}-1} dt, \quad (\text{V.3})$$

where  $k = n - p$  is the codimension.

In the case of codimension 1, this equation becomes:

$$I(g) = 2 \nu_{p-1} \int_0^1 \frac{(1-t^2)^{\frac{p}{2}-1}}{(1-r^2(1-t^2))^{1/2}} dt,$$

as wanted. In particular, the upper and lower bounds are given by

$$2\nu_{p-1} \int_0^1 (1-t^2)^{\frac{p}{2}-1} dt \leq I(g) \leq \frac{2\nu_{p-1}}{(1-r^2)^{1/2}} \int_0^1 (1-t^2)^{\frac{p}{2}-1} dt,$$

which yields

$$\nu_{p-1} B\left(\frac{1}{2}, \frac{p}{2}\right) \leq I(g) \leq \frac{\nu_{p-1} B\left(\frac{1}{2}, \frac{p}{2}\right)}{d_{\mathbb{P}}(g, S^p)},$$

as wanted.

In the case of even codimension  $k = n - p = 2\tau$ , with  $\tau \in \mathbb{N}^*$ , equation (V.3) yields:

$$I(g) = \frac{2\nu_{p-1}}{d_{\mathbb{P}}(g, S^p)} \sum_{i=0}^{\tau-1} \int_0^1 \binom{\tau-1}{i} \left(\frac{t^2}{s^2}\right)^i (1-t^2)^{\tau-i+\frac{p}{2}-2} dt.$$

Then,

$$I(g) = \sum_{i=0}^{\tau-1} \binom{\tau-1}{i} \frac{2\nu_{p-1}}{d_{\mathbb{P}}(g, S^p)^{2i+1}} \int_0^1 t^{2i} (1-t^2)^{\frac{n}{2}-i-2} dt,$$

and

$$I(g) = \sum_{i=0}^{\tau-1} \frac{\nu_{p-1}}{d_{\mathbb{P}}(g, S^p)^{2i+1}} \frac{B\left(i+\frac{1}{2}, \frac{n}{2}-i-1\right)}{\tau B\left(\tau-i, i+1\right)}.$$

In the case of odd codimension  $k = n - p = 2\tau + 1$ , with  $\tau \in \mathbb{N}^*$ , equation (V.3) yields:

$$I(g) = \frac{2\nu_{p-1}}{d_{\mathbb{P}}(g, S^p)} \int_0^1 \left( (1-t^2) + \frac{t^2}{s^2} \right)^{\tau-\frac{1}{2}} (1-t^2)^{\frac{p}{2}-1} dt.$$

Observing that

$$\frac{t}{s} \leq \left( (1-t^2) + \frac{t^2}{s^2} \right)^{1/2} = \left( \frac{s^2 + r^2 t^2}{s^2} \right)^{1/2} \leq \frac{1}{s}, \quad (\text{V.4})$$

equation (V.3) becomes:

$$I(g) = \frac{2\nu_{p-1}}{d_{\mathbb{P}}(g, S^p)} \int_0^1 \left( (1-t^2) + \frac{t^2}{s^2} \right)^{(\tau-1)+\frac{1}{2}} (1-t^2)^{\frac{p}{2}-1} dt.$$

Then, expanding  $\left( (1-t^2) + \frac{t^2}{s^2} \right)^{\tau-1}$  yields

$$I(g) = \sum_{i=0}^{\tau-1} \frac{2\nu_{p-1}}{d_{\mathbb{P}}(g, S^p)^{2i+1}} \binom{\tau-1}{i} \int_0^1 t^{2i} (1-t^2)^{\frac{n}{2}-i-\frac{5}{2}} \left( (1-t^2) + \frac{t^2}{s^2} \right)^{1/2} dt,$$

where

$$\binom{\tau-\frac{1}{2}}{i} = \frac{\left(\tau-\frac{1}{2}\right)_i}{i!} = \frac{1}{\left(\tau+\frac{1}{2}\right) B\left(\tau-i+\frac{1}{2}, i+1\right)},$$

and  $\left(\tau-\frac{1}{2}\right)_i$  is Pochhammer symbol:

$$\binom{\tau-\frac{1}{2}}{i} = \frac{\Gamma\left(\tau+\frac{1}{2}\right)}{\Gamma\left(\tau-i+\frac{1}{2}\right)}.$$

Thus,

$$I(g) \leq \sum_{i=0}^{\tau-1} \frac{2\nu_{p-1}}{d_{\mathbb{P}}(g, S^p)^{2i+2}} \frac{\int_0^1 t^{2i} (1-t^2)^{\frac{n}{2}-i-\frac{5}{2}} dt}{\tau B\left(\tau-i, i+1\right)},$$

and

$$I(g) \geq \sum_{i=0}^{\tau-1} \frac{2\nu_{p-1}}{d_{\mathbb{P}}(g, S^p)^{2i+2}} \frac{\int_0^1 t^{2i+1} (1-t^2)^{\frac{n}{2}-i-\frac{5}{2}} dt}{\tau B(\tau-i, i+1)}.$$

Namely,

$$I(g) \leq \sum_{i=0}^{\tau-1} \frac{2\nu_{p-1}}{d_{\mathbb{P}}(g, S^p)^{2i+2}} \frac{B(i+\frac{1}{2}, \frac{n}{2}-i-\frac{3}{2})}{\tau B(\tau-i, i+1)},$$

and

$$I(g) \geq \sum_{i=0}^{\tau-1} \frac{2\nu_{p-1}}{d_{\mathbb{P}}(g, S^p)^{2i+2}} \frac{B(i+1, \frac{n}{2}-i-\frac{3}{2})}{\tau B(\tau-i, i+1)}. \quad \square$$

**Remark V.33.** One may want a close formula for the latter case. In that case, we have to be careful when expanding equation (V.3) as we have to distinguish both cases when  $1-t^2 \geq \frac{t^2}{s^2}$  and when  $1-t^2 \leq \frac{t^2}{s^2}$ . Hence

$$I(g) = \frac{2\nu_{p-1}}{d_{\mathbb{P}}(g, S^p)} \sum_{i=0}^{\infty} \binom{\tau-\frac{1}{2}}{i} \times \left( \int_0^{\frac{1}{\sqrt{1+s^2}}} \left(\frac{t^2}{s^2}\right)^i (1-t^2)^{\frac{n}{2}-i-2} dt + \int_{\frac{1}{\sqrt{1+s^2}}}^1 \left(\frac{t^2}{s^2}\right)^{\frac{n-p}{2}-i-1} (1-t^2)^{\frac{p}{2}+i-1} dt \right).$$

This yields

$$I(g) = 2\nu_{p-1} \sum_{i=0}^{\infty} \binom{\tau-\frac{1}{2}}{i} \times \left( \int_0^{\frac{1}{\sqrt{1+s^2}}} \frac{t^{2i} (1-t^2)^{\frac{n}{2}-i-2} dt}{d_{\mathbb{P}}(g, S^p)^{2i+1}} + \int_{\frac{1}{\sqrt{1+s^2}}}^1 \frac{t^{n-p-2i-2} (1-t^2)^{\frac{p}{2}+i-1} dt}{d_{\mathbb{P}}(g, S^p)^{n-p-2i-1}} \right).$$

and, hence,

$$I(g) = \frac{\nu_{p-1}}{\tau + \frac{1}{2}} \sum_{i=0}^{\infty} \frac{1}{B(\tau-i+\frac{1}{2}, i+1)} \times \left( \frac{B\left(\frac{1}{1+s^2}; i+\frac{1}{2}, \frac{n}{2}-i-1\right)}{d_{\mathbb{P}}(g, S^p)^{2i+1}} + \frac{B\left(\frac{s^2}{1+s^2}; \frac{p}{2}+i, \frac{n-p-1}{2}-i\right)}{d_{\mathbb{P}}(g, S^p)^{n-p-2i-1}} \right),$$

where  $B(x; a, b)$  is the incomplete Beta function:

$$B(x; a, b) = \int_0^x t^{a-1} (1-t)^{b-1} dt.$$

### 3.5.2 Proof of Remark V.24

This remark immediately follows from equation (V.2). Making the obvious change of variable, this equation yields:

$$I(g) = 2\nu_{p-1} \int_0^{1/s} (1+r^2 z^2)^{\frac{n-p-2}{2}} (1-s^2 z^2)^{\frac{p-2}{2}} dz.$$

And, by the standard definition of Appell's  $F_1$  hypergeometric function, we immediately obtain:

$$I(g) = \frac{\nu_{p-1}}{2 d_{\mathbb{P}}(g, S^p)} F_1\left(\frac{1}{2}, \frac{p+2-n}{2}, \frac{2-p}{2}, \frac{3}{2}; -\cot(d_R(g, S^p)), 1\right),$$

where  $\cot(d_R(g, S^p))$  is the cotangent of the Riemannian distance of  $g$  to  $S^p$ .  $\square$

## 4 Proof of the main results

### 4.1 Proof of Theorem V.5

As above, we assume  $M = \mathbb{R}^{p+1}$ ,  $S(M) = S^p$  and  $k = n - p$  the codimension of  $S^p$  in  $S^n$ . Let  $\varphi: S^n \rightarrow \mathbb{R}_+$  be an integrable function and let  $I$  be the quantity:

$$I = \int_{(g,f) \in S^n \times S^p} \left( \int_{L(g,f)} \varphi(h) dL_{(g,f)}(h) \right) dS^n dS^p,$$

where  $L_{(g,f)} \in \mathcal{L}$  is the great circle containing  $g$  and  $f$  and  $dL_{(g,f)}$  is the standard measure on the great circle.

Let  $\Phi: S^n \times S^p \setminus \text{Diag} \rightarrow \mathcal{L}_M$ , be the mapping discussed in Section 3 and given by  $\Phi(g, f) = L_{(g,f)} \in \mathcal{L}_M$ , where  $\mathcal{L}_M$  is the semialgebraic set of great circles in  $\mathcal{L}$  that intersect  $S^p$ . According to the Co-area formula (Theorem V.27) we have:

$$I = \int_{\mathcal{L}_M} \left( \int_{\Phi^{-1}(L)} \frac{\theta(g, f)}{\text{NJ}_{(g,f)} \Phi} d\Phi^{-1}(L) \right) d\mathcal{L}_M,$$

where

$$\theta(g, f) = \int_{L(g,f)} \varphi(h) dL_{(g,f)}(h).$$

Note that, for  $L \in \mathcal{L}_M$ , if  $L \cap S^p = \{\pm f\}$ , we have  $\Phi^{-1}(L) = L \times \{f\} \cup L \times \{-f\}$  and we conclude:

$$I = 2 \int_{\mathcal{L}_M} \left( \int_L \frac{\theta(g, f)}{\text{NJ}_{(g,f)} \Phi} dL \right) d\mathcal{L}_M.$$

Now, from Proposition V.21 we conclude:

$$I = 2 \int_{\mathcal{L}_M} \frac{\theta(g, f)}{\partial_M(L)} \left( \int_L \frac{d_{\mathbb{P}}(g, S(M))^{n-1}}{\partial_M(L)^{n-1}} dL \right) d\mathcal{L}_M.$$

Then, from Lemma V.29 we conclude that the inner integral is constant and independent of  $L$  and, hence, the following holds:

$$I = 2 B\left(\frac{n+2}{2}, \frac{1}{2}\right) \int_{\mathcal{L}_M} \frac{\theta(g, f)}{\partial_M(L)} d\mathcal{L}_M,$$

*i.e.*

$$I = 2 B\left(\frac{n+2}{2}, \frac{1}{2}\right) \int_{\mathcal{L}_M} \left( \frac{1}{\partial_M(L)} \int_L \varphi(h) dL(h) \right) d\mathcal{L}_M.$$

Now, considering the incidence variety  $\mathcal{IC}(M)$  given by

$$\mathcal{IC}(M) = \{([A], g) \in \mathcal{L} \times S^n, g \in \text{Span}(A), [A] \in \mathcal{L}_M\},$$

and the canonical projections  $p_1: \mathcal{IC}(M) \rightarrow \mathcal{L}_M$  and  $p_2: \mathcal{IC}(M) \rightarrow S^n$ , and applying twice the Co-area formula allows to conclude:

$$I = 2 \text{B}\left(\frac{n+2}{2}, \frac{1}{2}\right) \int_{(L,g) \in \mathcal{IC}(M)} \left( \frac{\text{NJ}_{(L,g)} p_1}{\partial_M(L)} \varphi(g) \right) d\mathcal{IC}(M),$$

and,

$$I = 2 \text{B}\left(\frac{n+2}{2}, \frac{1}{2}\right) \int_{S^n} \left( \int_{p_2^{-1}(g)} \frac{1}{\partial_M(L)} \varphi(g) \frac{\text{NJ}_{(L,g)} p_1}{\text{NJ}_{(L,g)} p_2} d p_2^{-1}(g)(L) \right) d S^n.$$

Namely, we have:

$$I = 2 \text{B}\left(\frac{n+2}{2}, \frac{1}{2}\right) \int_{S^n} \varphi(g) \left( \int_{\mathcal{IC}(M)_g} \frac{1}{\partial_M(L)} \frac{\text{NJ}_{(L,g)} p_1}{\text{NJ}_{(L,g)} p_2} d\mathcal{IC}(M)_g(L) \right) d S^n.$$

According to the notation used in Proposition V.23, this equality may be rewritten as:

$$I = 2 \text{B}\left(\frac{n+2}{2}, \frac{1}{2}\right) \int_{S^n} \varphi(g) I(g) d S^n.$$

This proposition implies the following cases according to the codimension  $k = n - p$ :

- If  $k = 1$ , the following inequalities result from Proposition V.23:

$$I \geq 2 \nu_{p-1} \text{B}\left(\frac{1}{2}, \frac{p}{2}\right) \text{B}\left(\frac{n+2}{2}, \frac{1}{2}\right) \int_{S^n} \varphi(g) d S^n,$$

and

$$I \leq 2 \nu_{p-1} \text{B}\left(\frac{1}{2}, \frac{p}{2}\right) \text{B}\left(\frac{n+2}{2}, \frac{1}{2}\right) \int_{S^n} \frac{\varphi(g)}{d_{\mathbb{P}}(g, S^p)} d S^n.$$

As  $E$  is an expectation, we have

$$E = \frac{1}{\nu_n \nu_p} I,$$

and hence the following two inequalities:

$$\begin{aligned} E &\geq \frac{2 \nu_{p-1} \text{B}\left(\frac{1}{2}, \frac{p}{2}\right) \text{B}\left(\frac{n+2}{2}, \frac{1}{2}\right)}{\nu_p} \frac{1}{\nu_n} \int_{S^n} \varphi(g) d S^n, \\ E &\leq \frac{2 \nu_{p-1} \text{B}\left(\frac{1}{2}, \frac{p}{2}\right) \text{B}\left(\frac{n+2}{2}, \frac{1}{2}\right)}{\nu_p \text{B}\left(1, \frac{n-2}{2}\right)} \frac{\text{B}\left(1, \frac{n-2}{2}\right)}{\nu_n} \int_{S^n} \frac{\varphi(g)}{d_{\mathbb{P}}(g, S^p)} d S^n. \end{aligned}$$

According to Definition V.3, these two inequalities may be rewritten as

$$C(n, p) E_{S^n}[\varphi] \leq E \leq D(n, p) \mathbf{R}^{n-p-1} \varphi(S^p),$$

where

$$C(n, p) = \frac{2 \nu_{p-1} \text{B}\left(\frac{1}{2}, \frac{p}{2}\right) \text{B}\left(\frac{n+2}{2}, \frac{1}{2}\right)}{\nu_p},$$

and

$$D(n, p) = \frac{C(n, p)}{B(1, \frac{n-2}{2})} = \frac{n-2}{2} C(n, p).$$

Using Gautschi's [Gau60] and Kershaw's [Ker83] inequalities we conclude:

$$4 \sqrt{\frac{\pi(2p+1)}{(p+\sqrt{3}-2)(n+\sqrt{3})}} \leq C(n, p) \leq 4 \sqrt{\frac{\pi(p+\sqrt{3}-1)}{(2p-1)(n+3)}},$$

whereas

$$\frac{n-2}{2} \sqrt{\frac{\pi(2p+1)}{(p+\sqrt{3}-2)(n+\sqrt{3})}} \leq D(n, p) \leq \frac{n-2}{2} \sqrt{\frac{\pi(p+\sqrt{3}-1)}{(2p-1)(n+3)}},$$

- If  $k \in 2\mathbb{N}^*$  is an even integer number we have:

$$I = \sum_{i=0}^{\frac{k}{2}-1} 4 B\left(\frac{n+2}{2}, \frac{1}{2}\right) \nu_{p-1} \frac{B(i+\frac{1}{2}, \frac{n-i-1}{2})}{k B(\frac{k}{2}-i, i+1)} \int_{S^n} \frac{\varphi(g)}{d_{\mathbb{P}}(g, S^p)^{2i+1}} dS^n.$$

Namely, in terms of Definition V.3, we have proved

$$I = \sum_{i=0}^{\frac{k}{2}-1} \frac{4 B(\frac{n+2}{2}, \frac{1}{2}) \nu_{p-1} B(i+\frac{1}{2}, \frac{n-i-1}{2}) \nu_n}{k B(\frac{k}{2}-i, i+1)} \mathbf{R}^{k-2i-1} \varphi(S^p).$$

Namely, we have

$$E = \frac{1}{\nu_n \nu_p} I = \sum_{i=0}^{\frac{k}{2}-1} C(n, p, i) \mathbf{R}^{k-2i-1} \varphi(S^p),$$

where

$$C(n, p, i) = 2 \binom{\frac{n-p}{2}-1}{i} \frac{B(\frac{n+2}{2}, \frac{1}{2})}{B(\frac{p-1}{2}, \frac{1}{2})}.$$

- If  $k \in (2\mathbb{N}^* + 1)$  is an odd integer, according to Proposition V.23 we may use the finite sum bounds to conclude:

$$E \leq \sum_{i=0}^{\frac{k-3}{2}} \binom{\frac{k-3}{2}}{i} \frac{4 \nu_{p-1} B(\frac{n+2}{2}, \frac{1}{2}) B(i+\frac{1}{2}, \frac{n-2i-3}{2})}{\nu_p \nu_n} \int_{S^n} \frac{\varphi(g)}{d_{\mathbb{P}}(g, S^p)^{2i+2}} dS^n.$$

On the other hand the same proposition also yields:

$$E \geq \sum_{i=0}^{\frac{k-3}{2}} \binom{\frac{k-3}{2}}{i} \frac{4 \nu_{p-1} B(\frac{n+2}{2}, \frac{1}{2}) B(i+1, \frac{n-2i-3}{2})}{\nu_p \nu_n} \int_{S^n} \frac{\varphi(g)}{d_{\mathbb{P}}(g, S^p)^{2i+2}} dS^n.$$

Thus, we conclude

$$\sum_{i=0}^{\frac{k-3}{2}} A_1(n, p, i) \mathbf{R}^{k-2i-2} \varphi(S^p) \leq E \leq \sum_{i=0}^{\frac{k-3}{2}} A_2(n, p, i) \mathbf{R}^{k-2i-2} \varphi(S^p),$$

where

$$A_1(n, p, i) = 2 \binom{\frac{k-3}{2}}{i} \frac{(n-2) \text{B}(\frac{n+2}{2}, \frac{1}{2}) \Gamma(i+1)}{\text{B}(\frac{p-1}{2}, \frac{1}{2}) \Gamma(i+\frac{3}{2})},$$

and

$$A_2(n, p, i) = 4 \binom{\frac{k-3}{2}}{i} \frac{\text{B}(\frac{n+2}{2}, \frac{1}{2}) \Gamma(i+\frac{1}{2}) \Gamma(\frac{n}{2}-1)}{\text{B}(\frac{p-1}{2}, \frac{1}{2}) \Gamma(i+\frac{3}{2}) \Gamma(\frac{n}{2})}.$$

Now, using Gautschi's [Gau60] and Kershaw's [Ker83] inequalities, we conclude:

$$A_1(n, p, i) \geq \binom{\frac{n-p-3}{2}}{i} \frac{\text{B}(\frac{n+2}{2}, \frac{1}{2})}{\text{B}(\frac{p-1}{2}, \frac{1}{2})} \frac{2\sqrt{2}(n-2)}{\sqrt{2i+\sqrt{3}}} = \frac{B_0(n, p, i)(n-2)}{\sqrt{i+\sqrt{3}/2}}.$$

$$A_2(n, p, i) = 16 \binom{\frac{n-p-3}{2}}{i} \frac{\text{B}(\frac{n+2}{2}, \frac{1}{2})}{\text{B}(\frac{p-1}{2}, \frac{1}{2})} \frac{1}{(2i+1)(n-2)} = \frac{8 B_0(n, p, i)}{(2i+1)(n-2)}. \quad \square$$

## 4.2 Proof of Corollary V.7

With the same notation as above, we make use of inequalities (V.4) to conclude from equation (V.3):

$$\frac{2\nu_{p-1}}{\text{d}_{\mathbb{P}}(g, S^p)^{k-1}} \int_0^1 t^{k-2} (1-t^2)^{\frac{p}{2}-1} dt \leq I(g) \leq \frac{2\nu_{p-1}}{\text{d}_{\mathbb{P}}(g, S^p)^{n-p-1}}.$$

Namely,

$$\frac{\nu_{p-1} \text{B}(\frac{n-p}{2}, \frac{p}{2})}{\text{d}_{\mathbb{P}}(g, S^p)^{k-1}} \leq I(g) \leq \frac{2\nu_{p-1}}{\text{d}_{\mathbb{P}}(g, S^p)^{k-1}}.$$

From the proof of Theorem V.5 above, we conclude

$$E \geq \frac{2 \text{B}(\frac{n+2}{2}, \frac{1}{2}) \nu_{p-1} \text{B}(\frac{n-p}{2}, \frac{p}{2})}{\nu_p \nu_n} \int_{S^n} \frac{\varphi(g)}{\text{d}_{\mathbb{P}}(g, S^p)^{k-1}} dS^n,$$

and

$$E \leq \frac{2 \text{B}(\frac{n+2}{2}, \frac{1}{2}) \nu_{p-1}}{\nu_p \nu_n} \int_{S^n} \frac{\varphi(g)}{\text{d}_{\mathbb{P}}(g, S^p)^{k-1}} dS^n.$$

According to Definition V.3, this means:

$$E \geq \frac{2 \text{B}(\frac{n+2}{2}, \frac{1}{2}) \nu_{p-1}}{\nu_p} \mathbf{R}^1 \varphi(S^p),$$

and

$$E \leq \frac{2 \text{B}(\frac{n+2}{2}, \frac{1}{2}) \nu_{p-1}}{\nu_p \text{B}(\frac{n-p}{2}, \frac{p}{2})} \mathbf{R}^1 \varphi(S^p).$$

Using Gautschi's [Gau60] and Kershaw's [Ker83] inequalities, we finally obtain:

$$\sqrt{\frac{2p+1}{2(n+\sqrt{3})}} \mathbf{R}^1 \varphi(S^p) \leq E \leq 2 \sqrt{\frac{2(p+\sqrt{3}-1)}{2n+3}} \frac{1}{\text{B}(\frac{n-p}{2}, \frac{p}{2})} \mathbf{R}^1 \varphi(S^p),$$

as wanted.  $\square$

### 4.3 Proof of Proposition V.8

With the same notation as in the Introduction, according to Lemma V.29, for every  $L \in \mathcal{L}_M$ , we have:

$$E_{L_M}[\varphi] = \frac{1}{\mathbb{B}\left(\frac{n-p+2}{2}, \frac{1}{2}\right) \partial_M(L)^{n-p-1}} \int_L \varphi(g) \, d_{\mathbb{P}}(g, S^p)^{n-p-1} \, dL.$$

Then, we use the Co-area formula (Theorem V.27) as in the proof of Theorem V.5 above, to conclude:

$$\begin{aligned} \int_{\mathcal{L}_M} E_{L_M}[\varphi] &= \int_{S^n} \frac{\varphi(g)}{\mathbb{B}\left(\frac{n-p+2}{2}, \frac{1}{2}\right)} \\ &\quad \times \left( \int_{\mathcal{IC}(M)} \frac{d_{\mathbb{P}}(f, S^p)^{n-p-1}}{\partial_M(L)^{n-p-1}} \frac{\text{NJ}_{(L,g)} p_1}{\text{NJ}_{(L,g)} p_1} d[p_2^{-2}(g)](L) \right) dS^n(g) \end{aligned}$$

According to Proposition V.22, this yields:

$$\int_{\mathcal{L}_M} E_{L_M}[\varphi] = \int_{S^n} \frac{\varphi(g)}{\mathbb{B}\left(\frac{n-p+2}{2}, \frac{1}{2}\right)} \left( \int_{\mathcal{IC}(M)_g} d[p_2^{-2}(g)](L) \right) dS^n(g).$$

Then, applying Proposition V.31 we conclude:

$$\int_{\mathcal{L}_M} E_{L_M}[\varphi] = \frac{\nu_p}{\mathbb{B}\left(\frac{n-p+2}{2}, \frac{1}{2}\right)} \int_{S^n} \varphi(g) \, dS^n(g) = \frac{\nu_p \nu_n}{\mathbb{B}\left(\frac{n-p+2}{2}, \frac{1}{2}\right)} E_{S^n}[\varphi].$$

Now, taking  $\varphi = 1$ , we conclude:

$$\text{vol}[\mathcal{L}_M] = \frac{\nu_p \nu_n}{\mathbb{B}\left(\frac{n-p+2}{2}, \frac{1}{2}\right)} E_{S^n}[1] = \frac{\nu_p \nu_n}{\mathbb{B}\left(\frac{n-p+2}{2}, \frac{1}{2}\right)},$$

and Proposition V.8 follows immediately.  $\square$

## 5 Proof of the statements related to polynomial equation solving.

We follow the notation introduced in Section 1.3. We will use the notation  $\mathbb{S}^{2N+1}$  to denote  $\mathbb{S}(\mathcal{H}_{(d)})$  and  $\mathbb{S}^p$  to denote  $\mathbb{S}(M)$ . As in [SS93a, SS93b], let  $V_{(d)} \subseteq \mathbb{S}^{2N+1} \times \mathbb{P}^n(\mathbb{C})$  be the solution variety. Namely,

$$V_{(d)} = \{(f, \zeta) \in \mathbb{S}^{2N+1} \times \mathbb{P}^n(\mathbb{C}), \zeta \in V(f)\}.$$

### 5.1 Proof of Corollary V.9

Let us define  $\tilde{\Sigma} \subseteq \mathcal{L}_M$  as the subset of all great circles  $L \in \mathcal{L}_M$  that intersect the discriminant variety  $\Sigma$ . As  $\dim(\Sigma \cap \mathbb{S}(M)) < \dim \mathbb{S}(M)$ , using the double fibration as in Section 2 above, we may conclude that the codimension of  $\tilde{\Sigma}$  in  $\mathcal{L}_M$  is at least 1 and, hence, it is a semialgebraic set of volume zero. Namely,

$$E_{\mathcal{L}_M}[\chi_{\tilde{\Sigma}}] = 0,$$

where  $E_{\mathcal{L}_M}$  means expectation in  $\mathcal{L}_M$  and  $\chi_{\tilde{\Sigma}}: \mathcal{L}_M \rightarrow \{0, 1\}$  is the characteristic function defined by  $\tilde{\Sigma}$ .

Let us define the mapping  $\Theta_{\tilde{\Sigma}}: V_{(d)} \rightarrow \mathbb{R}_+$  given by the following identity:

$$\Theta_{\tilde{\Sigma}}(g, \zeta) = E_{\mathbb{S}^p}[\mathcal{C}(f, g, \zeta)] = \frac{1}{\nu_p} \int_{\mathbb{S}^p} \chi_{\tilde{\Sigma}}(L_{(g,f)}) d\mathbb{S}^p,$$

where  $L_{(g,f)}$  is the great circle passing through  $g$  and  $f$ . Let  $\mathcal{G}_{(d)} \subseteq V_{(d)}$  be the strong questor set defined in [BP11a], endowed with its probability distribution. The probability that the algorithm outputs FAILURE is at most the expectation  $E_{\mathcal{G}_{(d)}}[\Theta_{\tilde{\Sigma}}]$ . By [BP11a, Theorem 7], the following equality holds:

$$E_{\mathcal{G}_{(d)}}[\Theta_{\tilde{\Sigma}}] = \frac{1}{\nu_{2N+1}} \int_{\mathbb{S}^{2N+1}} \frac{1}{\mathcal{D}} \sum_{\zeta \in V_{\mathbb{P}}(g)} \Theta_{\tilde{\Sigma}}(g, \zeta) d\mathbb{S}^{2N+1}.$$

Namely, this expectation satisfies:

$$E_{\mathcal{G}_{(d)}}[\Theta_{\tilde{\Sigma}}] = \frac{1}{\nu_{2N+1} \nu_p} \int_{\mathbb{S}^{2N+1} \times \mathbb{S}^p} \frac{1}{\mathcal{D}} \sum_{\zeta \in V_{\mathbb{P}}(g)} \chi_{\tilde{\Sigma}}(L_{(g,f)}) d\mathbb{S}^{2N+1} d\mathbb{S}^p.$$

In other terms,

$$E_{\mathcal{G}_{(d)}}[\Theta_{\tilde{\Sigma}}] = \frac{1}{\nu_{2N+1} \nu_p} \int_{\mathbb{S}^{2N+1} \times \mathbb{S}^p} \chi_{\tilde{\Sigma}}(L_{(g,f)}) d\mathbb{S}^{2N+1} d\mathbb{S}^p.$$

According to Proposition V.21 and the Co-area formula, we have:

$$E_{\mathcal{G}_{(d)}}[\Theta_{\tilde{\Sigma}}] = \frac{1}{\nu_{2N+1} \nu_p} \int_{\mathfrak{e}_M} \left( \int_{L_{(g,f)}} \chi_{\tilde{\Sigma}}(L_{(g,f)}) \frac{d_{\mathbb{P}}(g, \mathbb{S}^p)^{n-1}}{\partial_M(L_{(g,f)})^n} dL_{(g,f)} \right) d\mathfrak{e}_M.$$

Finally, as  $d_{\mathbb{P}}(g, \mathbb{S}^p) \leq \partial_M(L_{(g,f)})$  we have

$$0 \leq E_{\mathcal{G}_{(d)}}[\Theta_{\tilde{\Sigma}}] \leq \frac{2\pi}{\nu_{2N+1} \nu_p} \int_{\mathcal{L}_M} \chi_{\tilde{\Sigma}}(L_{(g,f)}) \frac{1}{\partial_M(L_{(g,f)})} d\mathcal{L}_M.$$

As  $\tilde{\Sigma}$  has zero measure in  $\mathcal{L}_M$ , we conclude  $E_{\mathcal{G}_{(d)}}[\Theta_{\tilde{\Sigma}}] = 0$  and the claim of Corollary V.9 follows.  $\square$

## 5.2 Proof of Corollaries V.10, V.11 and V.12

Again we use the same strategy based on [BP11a]. Let us define the mapping  $\Theta: V_{(d)} \rightarrow \mathbb{R}_+$  given by the following identity:

$$\Theta(g, \zeta) = E_{\mathbb{S}^p}[\mathcal{C}(f, g, \zeta)] = \frac{1}{\nu_p} \int_{\mathbb{S}^p} \mathcal{C}(f, g, \zeta) d\mathbb{S}^p,$$

where  $d\mathbb{S}^p$  is the volume form associated to the Riemannian structure of  $\mathbb{S}^N$  and  $\nu_p$  is the volume of  $\mathbb{S}^p$ .

Let  $\mathcal{G}_{(d)} \subseteq V_{(d)}$  be the strong questor set defined in [BP11a], endowed with its probability distribution. By [BP11a, Theorem 7], the following equality holds:

$$E_M[\text{Time}] = E_{\mathcal{G}_{(d)}}[\Theta] = \frac{1}{\nu_{2N+1}} \int_{\mathbb{S}^{2N+1}} \frac{1}{\mathcal{D}} \sum_{\zeta \in V_{\mathbb{P}}(g)} \Theta(g, \zeta) d\mathbb{S}^{2N+1}, \quad (\text{V.5})$$

where  $E$  denotes expectation,  $\mathcal{D} = \prod_{i=1}^n d_i$  is the Bézout number associated to the list  $(d) = (d_1, \dots, d_n)$ ,  $d \mathbb{S}^{2N+1}$  the volume form in  $\mathbb{S}^{2N+1}$  and  $\nu_{2N+1}$  the volume of this sphere.

Now observe that equation (V.5) may be rewritten as:

$$E_M = \frac{1}{\nu_{2N+1} \nu_p} \int_{\mathbb{S}^{2N+1} \times \mathbb{S}^p} \frac{1}{\mathcal{D}} \sum_{\zeta \in \mathbb{V}_{\mathbb{P}}(g)} \mathcal{C}(f, g, \zeta) d \mathbb{S}^{2N+1} d \mathbb{S}^p.$$

From the definition of  $\mu_{\text{av}}^2(g)$ , we immediately conclude:

$$E_M = \frac{1}{\nu_{2N+1} \nu_p} \int_{\mathbb{S}^{2N+1} \times \mathbb{S}^p} \left( \int_{L(g,f)} \mu_{\text{av}}^2(h) d L_{(g,f)} \right) d \mathbb{S}^{2N+1} d \mathbb{S}^p.$$

In other words,

$$E_M = E_{(g,f) \in \mathbb{S}^{2N+1} \times \mathbb{S}^p} \left[ \int_{L(g,f)} \mu_{\text{av}}^2(h) d L_{(g,f)}(h) \right].$$

Then, Corollary V.10 immediately follows from Theorem V.5, whereas Corollary V.11 immediately follows from Corollary V.7.

As for Corollary V.12, we apply the Co-area formula and Proposition V.21 to conclude:

$$E_M = \frac{1}{\nu_{2N+1} \nu_p} \int_{\mathcal{L}_M} \left( \int_{(g,f) \in \Phi^{-1}(L)} \frac{C(L_{(g,f)})}{\text{NJ}_{(g,f)} \Phi} d \Phi^{-1}(L) \right) d \mathcal{L}_M,$$

where

$$C(L_{(g,f)}) = \int_{L(g,f)} \mu_{\text{av}}^2(h) d L_{(g,f)}.$$

As  $L = L_{(g,f)}$ , using Proposition V.21 we conclude:

$$E_M = \frac{1}{\nu_{2N+1} \nu_p} \int_{\mathcal{L}_M} C(L) \left( \int_{\Phi^{-1}(L)} \frac{d_{\mathbb{P}}(g, \mathbb{S}(M))^{2N}}{\partial_M(L)^{2N+1}} d \Phi^{-1}(L) \right) d \mathcal{L}_M.$$

Namely,

$$E_M = \frac{1}{\nu_{2N+1} \nu_p} \int_{\mathcal{L}_M} \frac{C(L)}{\partial_M(L)} \left( \int_{\Phi^{-1}(L)} \frac{d_{\mathbb{P}}(g, \mathbb{S}(M))^{2N}}{\partial_M(L)^{2N}} d \Phi^{-1}(L) \right) d \mathcal{L}_M.$$

For great circles  $L \in \mathfrak{C}_M$ , this equals:

$$E_M = \frac{2}{\nu_{2N+1} \nu_p} \int_{\mathcal{L}_M} \frac{C(L)}{\partial_M(L)} \left( \int_L \frac{d_{\mathbb{P}}(g, \mathbb{S}(M))^{2N}}{\partial_M(L)^{2N}} d L \right) d \mathcal{L}_M.$$

Then, according to Lemma V.29, this yields:

$$E_M = \frac{2 \text{B}\left(\frac{2N+3}{2}, \frac{1}{2}\right)}{\nu_{2N+1} \nu_p} \int_{\mathcal{L}_M} \frac{C(L)}{\partial_M(L)} d \mathcal{L}_M.$$

According to Proposition V.8, this equality becomes:

$$E_M = \frac{2 \text{B}\left(N + \frac{3}{2}, \frac{1}{2}\right)}{\text{B}\left(N + 1 - \frac{p}{2}, \frac{1}{2}\right)} \frac{1}{\text{vol}[\mathcal{L}_M]} \int_{\mathcal{L}_M} \frac{1}{\partial_M(L)} \left( \int_L \mu_{\text{av}}^2(h) d L \right) d \mathcal{L}_M.$$

Namely, we proved

$$E_M = T(N, p) E_{\mathcal{L}_M} \left[ \frac{1}{\partial_M(L)} \int_L \mu_{\text{av}}^2(h) \, dL \right],$$

where

$$T(N, p) = \frac{2 \, \text{B}\left(N + \frac{3}{2}, \frac{1}{2}\right)}{\text{B}\left(N + 1 - \frac{p}{2}, \frac{1}{2}\right)},$$

and Corollary V.12 follows.  $\square$

## Bibliography

- [AG90] E. L. Allgower and K. Georg. *Numerical continuation methods*, volume 13 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 1990. An introduction.
- [AW05] J.-M. Azaïs and M. Wschebor. On the Roots of a Random System of Equations. The Theorem of Shub and Smale and Some Extensions. *Found. Comput. Math.*, 5(2):125–144, 2005.
- [BC11] P. Bürgisser and F. Cucker. On a problem posted by Steve Smale. *Ann. of Math. (2)*, 174(3):1785–1836, 2011.
- [BCL06] P. Bürgisser, F. Cucker and M. Lotz. Smoothed analysis of complex conic condition numbers. *J. Math. Pures Appl. (9)*, 86(4):293–309, 2006.
- [BCR98] J. Bochnak, M. Coste and M.-F. Roy. *Real Algebraic Geometry*, volume 36 of *Ergebnisse der Mathematik und ihrer Grenzgebiete (3) [Results in Mathematics and Related Areas (3)]*. Springer-Verlag, Berlin, 1998. Translated from the 1987 French original, Revised by the authors.
- [BCSS98] L. Blum, F. Cucker, M. Shub and S. Smale. *Complexity and real computation*. Springer-Verlag, New York, 1998. With a foreword by Richard M. Karp.
- [BDMS09] C. Beltrán, J.-P. Dedieu, G. Malajovich and M. Shub. Convexity properties of the condition number. *SIAM J. Matrix Anal. Appl.*, 31(3):1491–1506, 2009.
- [Bel11] C. Beltrán. A continuation method to solve polynomial systems and its complexity. *Numer. Math.*, 117(1):89–113, 2011.
- [BGH+10] B. Bank, M. Giusti, J. Heintz, M. Safey El Din and É. Schost. On the geometry of polar varieties. *Appl. Algebra Engrg. Comm. Comput.*, 21(1):33–83, 2010.
- [BGH+12] B. Bank, G. Giusti, J. Heintz, L. Lehmann and L. M. Pardo. Algorithms of Intrinsic Complexity for Point Searching in Compact Real Singular Hypersurfaces. *Foundations of Computational Mathematics*, 12:75–122, 2012.
- [BGHP05] B. Bank, M. Giusti, J. Heintz and L. M. Pardo. Generalized polar varieties: geometry and algorithms. *J. Complexity*, 21(4):377–412, 2005.
- [BGHP09] B. Bank, M. Giusti, J. Heintz and L. M. Pardo. On the intrinsic complexity of point finding in real singular hypersurfaces. *Inform. Process. Lett.*, 109(19):1141–1144, 2009.
- [Bor11] C. E. Borges. *Programación Genética, Algoritmos Evolutivos y Aprendizaje Inductivo: Hacia una Solución al Problema XVII de Smale en el Caso Real*. PhD thesis, Univ. Cantabria, 2011.
- [BP06] C. Beltrán and L. M. Pardo. On the complexity of non universal polynomial equation solving: old and new results. In *Foundations of Computational Mathematics, Santander 2005*, volume 331 of *London Math. Soc. Lecture Note Ser.*, pages 1–35. Cambridge Univ. Press, Cambridge, 2006.

- [BP08] C. E. Borges and L. M. Pardo. On the probability distribution of data at points in real complete intersection varieties. *J. Complexity*, 24(4):492–523, 2008.
- [BP09a] C. Beltrán and L. M. Pardo. Efficient polynomial system-solving by numerical methods. *J. Fixed Point Theory Appl.*, 6(1):63–85, 2009.
- [BP09b] C. Beltrán and L. M. Pardo. Smale’s 17th problem: Average polynomial time to compute affine and projective solutions. *J. Amer. Math. Soc.*, 22(2):363–385, 2009.
- [BP11a] C. Beltrán and L. M. Pardo. Fast linear homotopy to find approximate zeros of polynomial systems. *Found. Comput. Math.*, 11(1):95–129, 2011.
- [BP11b] J. Berthomieu and L. M. Pardo. Spherical Radon transform and the average of the condition number on certain Schubert subvarieties of a Grassmannian. Manuscript to appear in *J. Complexity*, 2011.
- [BPR06] S. Basu, R. Pollack and M.-F. Roy. *Algorithms in Real Algebraic Geometry*, volume 10 of *Algorithms and Computation in Mathematics*. Springer-Verlag, Berlin, Second edition, 2006.
- [BS09] C. Beltrán and M. Shub. Complexity of Bézout’s theorem. VII. Distance estimates in the condition metric. *Found. Comput. Math.*, 9(2):179–195, 2009.
- [BS11] D. Bates and F. Sottile. Khovanskii–Rolle continuation for real solutions. *Foundations of Computational Mathematics*, 11:563–587, 2011. 10.1007/s10208-011-9097-1.
- [Cho00] K.-K. Choi. On the distribution of points in projective space of bounded height. *Trans. Amer. Math. Soc.*, 352(3):1071–1111, 2000.
- [CKMW08] F. Cucker, T. Krick, G. Malajovich and M. Wschebor. A numerical algorithm for zero counting. I. Complexity and accuracy. *J. Complexity*, 24(5-6):582–605, 2008.
- [CKMW09] F. Cucker, T. Krick, G. Malajovich and M. Wschebor. A numerical algorithm for zero counting. II. Distance to ill-posedness and smoothed analysis. *J. Fixed Point Theory Appl.*, 6(2):285–294, 2009.
- [CKMW10] F. Cucker, T. Krick, G. Malajovich and M. Wschebor. A numerical algorithm for zero counting. III. Randomization and Condition. Manuscript, 2010.
- [Dem89] M. Demazure. *Catastrophes et bifurcations*. Les cours de l’École polytechnique. Ellipses, Paris, 1989.
- [DMS11] J.-P. Dedieu, G. Malajovich and M. Shub. Adaptive step size selection for homotopy methods to solve polynomial equations. Manuscript available from <http://arxiv.org/abs/1104.2084>, 2011.
- [DY93] J.-P. Dedieu and J.-C. Yakoubsohn. Computing the real roots of a polynomial by the exclusion algorithm. *Numer. Algorithms*, 4(1-2):1–24, 1993.
- [Fed69] H. Federer. *Geometric measure theory*. Die Grundlehren der mathematischen Wissenschaften, Band 153. Springer-Verlag New York Inc., New York, 1969.
- [Gau60] W. Gautschi. Some elementary inequalities relating to the gamma and incomplete gamma function. *J. Math. and Phys.*, 38:77–81, 1959/1960.
- [Ker83] D. Kershaw. Some extensions of W. Gautschi’s Inequalities for the Gamma Function. *Math. Comp.*, 41(164):607–611, 1983.
- [Li03] T. Y. Li. Numerical solution of polynomial systems by homotopy continuation methods. In *Handbook of numerical analysis, Vol. XI*, Handb. Numer. Anal., XI, pages 209–304. North-Holland, Amsterdam, 2003.
- [McL02] A. McLennan. The expected number of real roots of a multihomogeneous system of polynomial equations. *Amer. J. Math.*, 124(1):49–73, 2002.
- [Mor87] A. Morgan. *Solving polynomial systems using continuation for engineering and scientific problems*. Prentice Hall Inc., Englewood Cliffs, NJ, 1987.

- [Mor09] F. Morgan. *Geometric measure theory*. Elsevier/Academic Press, Amsterdam, Fourth edition, 2009. A beginner's guide.
- [Rub02] B. Rubin. Inversion formulas for the spherical Radon transform and the generalized cosine transform. *Adv. in Appl. Math.*, 29(3):471–497, 2002.
- [San04] L. A. Santaló. *Integral geometry and geometric probability*. Cambridge Mathematical Library. Cambridge University Press, Cambridge, Second edition, 2004. With a foreword by Mark Kac.
- [Shu09] M. Shub. Complexity of Bézout's theorem. VI. Geodesics in the condition (number) metric. *Found. Comput. Math.*, 9(2):171–178, 2009.
- [Sma00] S. Smale. Mathematical problems for the next century. In *Mathematics: frontiers and perspectives*, pages 271–294. Amer. Math. Soc., Providence, RI, 2000.
- [SS93a] M. Shub and S. Smale. Complexity of Bézout's theorem. I. Geometric aspects. *J. Amer. Math. Soc.*, 6(2):459–501, 1993.
- [SS93b] M. Shub and S. Smale. Complexity of Bézout's theorem. II. Volumes and probabilities. In *Computational algebraic geometry (Nice, 1992)*, volume 109 of *Progr. Math.*, pages 267–285. Birkhäuser Boston, Boston, MA, 1993.
- [SS96] M. Shub and S. Smale. Complexity of Bézout's theorem. IV. Probability of success; Extensions. *SIAM J. Numer. Anal.*, 33(1):128–148, 1996.
- [SW05] A. J. Sommese and C. W. Wampler, II. *The numerical solution of systems of polynomials*. World Scientific Publishing Co. Pte. Ltd., Hackensack, NJ, 2005. Arising in engineering and science.
- [Wsc05] M. Wschebor. On the Kostlan–Shub–Smale model for random polynomial systems. Variance of the number of roots. *J. Complexity*, 21(6):773–789, 2005.



# Annexe

## Introduction (translated into English)

Algebraic system solving is a problem at the heart of algebra and in particular of algebraic geometry. Its applications in mathematics and in industry are multiple. Historically and technically, this problem is inseparable from multiplicities handling. This thesis is about some particular aspects linked to the efficiency of such a resolution. Firstly, in Chapter I, we deal with minimizing the number of additive variables in such a system. We mention applications in cryptography wherein some systems, *a priori* underdetermined, must be solved in positive characteristic. Then, in Chapter II, we propose fast arithmetic routines for computing the product, the quotient and more generally the fixed point of a function defined over  $\mathbb{Q}_p$ . We use these routines in Chapter III to solve some systems, which can be well evaluated, locally in  $\mathbb{Q}_p$ . In Chapter IV, we consider the plane curves decomposition into irreducible components when the size of the Newton polygon is small compared to the product of the partial degrees. Finally, in Chapter V, we want to find an approximate complex root of a system with coefficients in  $\mathbb{R}$ .

### a Minimization of the number of additive variables

Chapter I is based on a joint work with P. Hivert and H. Mourtada titled *Computing Hironaka's invariants: Ridge and Directrix* and published in *Arithmetic, Geometry, Cryptography and Coding Theory 2009* [BHM10].

Let  $I$  be an ideal spanned by polynomials  $f_1, \dots, f_r \in \mathbb{K}[X_1, \dots, X_n]$ , the set  $V(I)$  of points  $x$  of  $\bar{\mathbb{K}}^n$  such that every polynomial of  $I$  vanishes at  $x$  is called the *affine variety defined by  $I$* , where  $\bar{\mathbb{K}}$  is the algebraic closure of  $\mathbb{K}$ . A *singular point* of the variety is a point  $x = (x_1, \dots, x_n) \in V(I)$  such that for every  $f \in I$ , the partial derivatives of  $f$  also vanish at  $x$ , *i.e.* such that

$$f(x_1, \dots, x_n) = \frac{\partial f}{\partial X_1}(x_1, \dots, x_n) = \dots = \frac{\partial f}{\partial X_n}(x_1, \dots, x_n) = 0.$$

If  $f$  is a bivariate polynomial in  $X$  and  $Y$ , then  $V(f)$  is a *plane curve*  $\mathcal{C}$ . The most simple singularities that  $\mathcal{C}$  can have are *nodes*, or self-intersection, and *cusps*, which are points wherein the curve only has half-tangents.

For example, the curve defined by  $f(X, Y) = Y^2 - X^3$  is singular at the origin, it is a cusp. Desingularization consists of finding a non singular curve  $\mathcal{C}'$  such that there exists a birational morphism from  $\mathcal{C}'$  to  $\mathcal{C}$ .

Since Hironaka's work in the 1960's and the 1970's, the desingularization problem is well-understood in characteristic zero (see [Hir64, Hir67, Hir70]). The positive characteristic case is however more difficult. For example, the desingularization of threefolds has only been done recently by Cossart and Piltant (see [CP08, CP09]). It should be noted that their proof is not constructive. The case of fourfolds is still an open problem. In [Hir64, Hir67], Hironaka introduces two invariants for the resolution of singularities: the *ridge* and the *directrix*. Given a homogeneous ideal  $I \subseteq \mathbb{K}[X_1, \dots, X_n]$  and a cone  $\mathcal{C} = \text{Spec } \mathbb{K}[X_1, \dots, X_n]/I$ , these two invariants are subsets of the tangent cone of the variety in the neighborhood  $x \in \mathcal{C}$ . In fact, according to Giraud [Gir75], the ridge is the tangent cone of a variety with a maximal contact with  $\mathcal{C}$  in a neighborhood of  $x$ . The directrix is a vector space, this is the biggest vector subspace  $W$  of  $\mathbb{A}^n = \text{Spec } \mathbb{K}[X_1, \dots, X_n]$  such that  $\mathcal{C} + W = \mathcal{C}$ . More formally, we have the following definition.

**Definition 1.** *Let  $\mathcal{C} = \text{Spec } \mathbb{K}[X_1, \dots, X_n]/I$  with  $I \subseteq \mathbb{K}[X_1, \dots, X_n]$ , a homogeneous ideal. The directrix of  $\mathcal{C}$  is the biggest free family  $(Y_1, \dots, Y_f)$ , where each  $Y_j$  is a linear form in the  $X_i$ , such that*

$$I = (I \cap \mathbb{K}[Y_1, \dots, Y_f]) \mathbb{K}[X_1, \dots, X_n].$$

*In other words, the directrix is the smallest set of needed variables to define  $I$ .*

An analogous definition exists for the ridge, which is just an additive subgroup of  $\mathbb{K}^n$ .

**Definition 2.** *The ridge of  $\mathcal{C}$  is the biggest additive subgroup of  $\mathbb{A}^n$  spanned by the smallest set of additive polynomials  $P_1, \dots, P_e$  such that*

$$I = (I \cap \mathbb{K}[P_1, \dots, P_e]) \mathbb{K}[X_1, \dots, X_n].$$

Since the notions of linear form and additive polynomial are the same in characteristic zero, it is clear that the ridge and the directrix are the same object in this case. Therefore, a maximal contact variety is always smooth. However, this is not true in positive characteristic  $p$ , so that a maximal contact variety can be not smooth. This obstacle prevents an extension of Hironaka's proof for desingularizing in positive characteristic. We deal with these two invariants of variety, the ridge and the directrix. We follow Giraud's work [Gir72, Gir75] wherein he introduces a functorial definition of the ridge.

**Proposition-Definition 3.** *Let  $\mathbb{A}_{\mathbb{K}}^n$  be the affine space over  $\mathbb{K}$  of dimension  $n$ . Let  $F$  be the functor mapping a  $\mathbb{K}$ -scheme  $S$  to the set  $F(S)$  of points  $v \in \mathbb{A}_{\mathbb{K}}^n$  such that for every  $S$ -point  $c \in \mathcal{C}(S)$ ,*

$$(v + c) \in \mathcal{C}(S).$$

*Then the functor  $F$  is representable by a scheme  $F$  that we call the ridge of  $\mathcal{C}$ .*

We have the following result.

**Corollary 4. (Corollary I.14, [BHM10, Corollary 2.12])** *Let  $U$  be the algebra of functions defined over  $\mathbb{A}_{\mathbb{K}}^n$  such that for every  $\mathbb{K}$ -scheme  $S$  and every  $S$ -point  $(u, v)$  in  $F \times_{\mathbb{K}} \mathbb{A}_{\mathbb{K}}^n$ , we have  $f(u + v) = f(u)$ .*

Let  $I$  be a homogeneous ideal of  $\mathbb{K}[X_1, \dots, X_n]$  and let  $\mathcal{G} := \{\gamma_1, \dots, \gamma_s\}$  be any homogeneous reduced Gröbner basis of  $J$  the ideal of the ridge of  $V(I)$ , then

$$I = (I \cap \mathbb{K}[\gamma_1, \dots, \gamma_s]) \mathbb{K}[X_1, \dots, X_n].$$

Furthermore,  $U = \mathbb{K}[\gamma_1, \dots, \gamma_s]$  and if  $K$  is a  $\mathbb{K}$ -algebra generated by additive polynomials such that

$$I = (I \cap K) \mathbb{K}[X_1, \dots, X_n],$$

then  $U \subset K$ .

Thus, Definitions 2 and 3 of the ridge coincide.

Furthermore, we give an algorithm for computing the ridge of an ideal. We also show that if  $(P_1, \dots, P_e)$  is a family spanning the ridge, then there exist integers  $\alpha_1, \dots, \alpha_e$  such that the family  $(\sqrt[p^{\alpha_1}]{P_1}, \dots, \sqrt[p^{\alpha_e}]{P_e})$  spans the directrix. If  $\mathbb{K}$  is a perfect field, then we can deduce an algorithm computing the directrix. Unfortunately, knowing if an element is a  $p^{\alpha}$ th power in characteristic  $p$  can be undecidable even if the considered ring is effective (see [FS56]).

## b Relaxed algorithms for $p$ -adic integers

Chapter II is an adaptation with J. van der Hoeven and G. Lecerf of the relaxed routines of formal power series for the  $p$ -adic integers. This work gave rise to an article *Relaxed algorithms for  $p$ -adic numbers* accepted for publication in *Journal de Théorie des Nombres de Bordeaux* [BHL11].

The *normalization* consists of solving codimension 1 singularities. In the case of a singular curve  $\mathcal{C}$ , singular loci are points, so they all have codimension 1. Thus, it suffices to normalize  $\mathcal{C}$  to find  $\mathcal{C}'$ , the desingularized of  $\mathcal{C}$ . If the characteristic of  $\mathbb{K}$  is 0, then normalizing  $\mathcal{C} = V(f)$  is equivalent to computing the *integral closure* of  $\mathbb{K}[X]$  in  $\mathbb{K}(X)[Y]/(f)$  (see [Sha94, Chapter II, Section 5]). Let  $R$  and  $R'$  be two rings such that  $R$  is a subring of  $R'$ ,  $b \in R'$  is *integral* over  $R$  if, equivalently [Lan02, Chapter VII],

- the subring  $R[b]$  is finitely generated over  $R$ ;
- there exist  $m \in \mathbb{N}^*$  and  $a_0, \dots, a_{m-1} \in R$  such that  $b$  is a root of the polynomial  $T^m + a_{m-1}T^{m-1} + \dots + a_0$ .

The integral closure  $\bar{R}$  of  $R$  in  $R'$  is the set of all  $b \in R'$  integral over  $R$ . Since the notion of being integral is stable by ring operations,  $\bar{R}$  is a subring of  $R'$  and a  $R$ -module.

**Example 5.** With the same example as above,  $f(X, Y) = Y^2 - X^3$ , it is clear that  $X^{3/2} = Y \in \mathbb{K}(X)[Y]/(f)$  is integral over  $\mathbb{K}[X]$ , so that the integral closure  $\overline{\mathbb{K}[X]}$  of  $\mathbb{K}[X]$  contains  $\mathbb{K}[X, X^{3/2}]$ . However, one can see that  $X^{1/2} = \frac{Y}{X}$  is a root of  $T^2 - X$ , therefore  $\mathbb{K}[X, X^{1/2}] = \mathbb{K}[X^{1/2}] \subseteq \overline{\mathbb{K}[X]}$ . Indeed, this not strict inclusion is in fact an equality.

When  $R$  is principal, the integral closure  $\bar{R}$  of  $R$  in a finite extension of  $K = \text{Quot } R$  is a free finitely generated  $R$ -module of rank  $n$ . Instead of computing a basis of  $\bar{R}$ , it is in general easier to compute a basis of its completion  $\bar{R}_p$  for the  $p$ -adic valuation in one of the localizations  $K_p$  of  $K$ , where  $p$  is a prime of  $R$ . Let us mention that such algorithms exist in [Hal01, Hoe94, Tra84]. It suffices to compute these *local* bases for  $p$  such that  $p^2 \mid \text{disc } f$ . The main advantage of computing local bases is that each element  $b_i$  of the basis  $(b_0, \dots, b_{n-1})$  can be written

$$b_i = \frac{b_{i,0} + \dots + b_{i,n-1} \alpha^{n-1}}{p^{d_i}},$$

where for all  $j$ ,  $0 \leq j \leq n-1$ ,  $b_{i,j} \in R_p$  and for all  $i$ ,  $\text{val}_p(b_i) \geq 0$ . Furthermore, we know a bound for the exponent  $d_i$ :

$$2 d_i \leq \text{val}_p(\text{disc } f).$$

In fact, one can improve this upper bound and get [Hoe94, Section 2.3]

$$2(d_0 + \dots + d_{n-1}) \leq \text{val}_p(\text{disc } f). \quad (1)$$

Then, one has to compute a *global* basis from all these local bases.

**Example 6.** Polynomial  $f(Y) = Y^2 - 5$  defines the extension  $\mathbb{Q}(\sqrt{5})$  over  $\mathbb{Q}$ . Since  $\sqrt{5}$  is integral over  $\mathbb{Z}$ , one knows that the integral closure of  $\mathbb{Z}$  contains  $\mathbb{Z}[\sqrt{5}]$ . As the discriminant of  $f$  is  $-4 = -2^2$ , only one computation of a local basis, namely the one in  $\mathbb{Q}_2(\sqrt{5})$ , suffices. According to equation (1), a basis  $(1, b_1)$  can at best be such that  $b_1 = \frac{1+b_{1,1}\sqrt{5}}{2}$  with  $b_{1,1} \in \mathbb{Z}_2$ . Over  $\mathbb{F}_2$ , polynomial  $f$  factors as

$$f(Y) = Y^2 + 1 = (Y + 1)^2,$$

therefore, the dyadic expansion of  $\sqrt{5}$  starts with 1, so that  $\text{val}_2\left(\frac{1+\sqrt{5}}{2}\right) = 0$ . One deduces that  $b_1 = \frac{1+\sqrt{5}}{2}$  is integral. Indeed, it is the golden ratio  $\varphi$ , root of  $T^2 - T - 1$ . There is only one local basis, so there is no problem to compute a global basis and the integral closure of  $\mathbb{Z}$  in  $\mathbb{Q}(\sqrt{5})$  is  $\mathbb{Z}[\varphi]$ .

The computation of an integral basis in dimension 1, that is the ring of integral elements over  $\mathbb{K}[X]$  in a *function field*  $\mathbb{K}(X)[Y]/(f)$  is very similar to the computation of an integral basis of the ring of integral elements in a *number field*  $\mathbb{Q}[Y]/(f)$ . We have seen above that the most common algorithms first compute some bases in some localizations to obtain then a global basis. For a function field, these localizations are the fields of Laurent series  $\mathbb{K}((X - \alpha))$ , with  $\alpha \in \mathbb{K}$  if  $\mathbb{K} = \bar{\mathbb{K}}$ , for a number field, these are the fields of  $p$ -adic numbers  $\mathbb{Q}_p$ , with  $p \in \mathbb{N}$  prime. For implementing some arithmetic routines over such objects, mainly two frameworks are available. The first one, the so-called *zealous* framework, consists of working with a fixed precision  $n$ , *i.e.* a power series  $S \in \mathbb{K}[[X]]$ ,

$$S = \sum_{k=0}^{\infty} S_k X^k,$$

will be represented by a truncated polynomial of degree  $n$ :

$$S = S_0 + \dots + S_{n-1} X^{n-1} + O(X^n).$$

The main advantage of this representation is that one can use all the fast routines available for computing the product of polynomials, in particular, the product of two series in precision  $n + 1$  can be computed in  $M(n) \in \tilde{O}(n)$  operations in the ground field  $\mathbb{K}$ . As a drawback, if during these computations, there is a loss of precision such that the result is not precised enough, one has to resume the computation from the beginning with increasing the needed precision.

Another way is to represent every computable series by a vector of computed coefficients  $(S_0, \dots, S_\ell)$  endowed with a `next( $n$ )` function capable of returning the coefficient  $S_n$  from the preceding ones. For example, the exponential computed up till precision 10 will be represented by vector  $(1, \dots, \frac{1}{9!})$  and by the `next( $n$ )` returning  $S_0 = 1$  if  $n = 0$  and  $S_n := \frac{S_{n-1}}{n}$  otherwise. Such a representation is called *lazy* because the coefficients are only computed when needed. It seems natural to define the `next( $n$ )` method of the product  $ST$  by the naive return of  $S_0 T_n + S_1 T_{n-1} + \dots + S_n T_0$ , unfortunately, this gives rise to a complexity quadratic in the precision for the product of two series:  $M(n) \in O(n^2)$ . However, the loss of precision is not a problem anymore, if one needs some few more coefficients, one may just call the `next( $n$ )` function as many times as needed without resuming the computation from the beginning. In 1997, and then in 2002, van der Hoeven proposed a fast product for lazy series called the *relaxed* product [Hoe97, Hoe02]. The main idea is to allow some unneeded computations at the time by doing products of polynomials of degree 1, 2, 4, ..., in order to profit from their complexities (see Figure II.1 in Chapter II). Compared to the zealous product, the overhead is at worst logarithmic, so that the product of two series up till precision  $n$  is computed in  $R(n) \in O(M(n) \log n)$  operations in  $\mathbb{K}$ .

We present an adaptation to all  $I$ -adic completions of a ring. The main motivation is the handling of the carry which can appear while doing an addition or a multiplication. In general, these rings contain  $\mathbb{Z}_p$  as a subring, that is why we deal in particular with the rings of  $p$ -adic integers. However, these are not the only ones.

**Example 7.** Let us complete the ring  $\mathbb{R}[X]$  for the  $(X^2 + 1)$ -adic valuation. As the ideal  $(X^2 + 1)$  is prime, the completion  $\mathbb{R}[X]_{(X^2+1)}$  is an integral domain and is the set of all elements  $S$  written

$$S = \sum_{k=0}^{\infty} (a_k X + b_k) (X^2 + 1)^k.$$

The sum of two series  $S$  and  $T$  is done component-wise, but the product can be the root of a carry. If  $S = 2X - 1$ , then

$$S^2 = (2X - 1)^2 = 4X^2 - 4X + 1 = -(4X + 3) + 4(X^2 + 1).$$

Let us notice, however, that as the residue field of  $\mathbb{R}[X]/(X^2 + 1)$  is  $\mathbb{R}[X]/(X^2 + 1) \simeq \mathbb{C}$ , then, according to [Coh46, Theorem 15], this ring is isomorphic to  $\mathbb{C}[[T]]$  wherein there is no carry handling. One of these isomorphisms send  $T$  onto  $X^2 + 1$  and  $i$  onto  $X + \frac{1}{2}X(X^2 + 1) + \frac{3}{8}X(X^2 + 1)^2 + \dots$ .

We show that the complexity bounds obtained for formal power series can be transposed to the  $p$ -adic integers. Let us denote  $l(m)$  the cost of the product of two integers whose bit-size is at most  $m$  bits and  $l_p(n)$  the cost of the product of two  $p$ -adic expansions of order  $n$  and whose coefficients have the usual binary representations.

**Proposition 8. (Propositions II.6 and II.7, [BHL11, Propositions 3.1 and 3.2])** *Let  $a$  and  $b$  be two relaxed  $p$ -adic integers. The product  $ab$  can be computed up till precision  $n$  using  $O(l_p(n) \log n)$  bit-operations. For this computation, the total amount of space needed does not exceed  $O(n)$ .*

*By making use of conversions from base 2 to base  $p$  and vice versa, the computation of  $a/b$  up till precision  $n$  can be done using  $O(l(n \log p) \log n)$  bit-operations and  $O(n \log p)$  bit-space.*

A recursive series is a power series  $S$  fixed point of a function  $\Phi$ , *i.e.* such that  $S = \Phi(S)$ . If, furthermore, the computation of the  $n$ th term  $\Phi(S)_n$  only needs to know  $S_0, \dots, S_{n-1}$ , for every  $n \geq k$ , then  $S$  has order  $k$  and from the  $k$  first terms of  $S$  and the function  $\Phi$ , one can compute recursively any term of  $S$ . If a series is invertible, then its inverse is a recursive series of order 1. We adapt some algorithms for recursive  $p$ -adic numbers, in particular, this of the computation of the inverse. We deduce that the complexity bounds of Proposition 8 are still valid for the computation of the quotient  $a/b$  of two  $p$ -adics integers up till precision  $n$ . In Section 7, we give a recursive method to compute the  $r$ th root in  $\mathbb{Z}_p$  of a  $p$ -adic integer. If  $r$  and  $p$  are coprime, Hensel's lemma ensures that such a root exists as soon as there exists a root modulo  $p$ .

## c Relaxed resolution of algebraic systems

Chapter III is a work in progress with R. Lebreton named *Relaxed  $p$ -adic Hensel lifting for algebraic systems* [BL12] extending the recursive computation of a  $r$ th root in  $\mathbb{Z}_p$ .

Let us consider systems with rational coefficients. Knowing if such systems have rational solutions can be complicated: the size of the numerators and of the denominators can be very big. Even by making use of fast routines over  $\mathbb{Q}$ , such as those provided by GMP [G+91], a non negligible overhead comes from the reduction to irreducible fractions. One can rather solve these systems in  $\mathbb{Q}_p$ , the  $p$ -adic completion of  $\mathbb{Q}$ , for one or multiple well-chosen  $p$ . Amongst all the computed solutions, one must determine those being susceptible to be in  $\mathbb{Q}$ .

**Example 9.** Let us consider the simplest case of a univariate polynomial:

$$f(X) = X^4 - 1.$$

Clearly, this polynomial only has two rational roots 1 and  $-1$ . Indeed, if  $p = 2$  or  $p \equiv 3 \pmod{4}$ , then  $f$  factors over  $\mathbb{Q}_p$  as

$$f(X) = (X - 1)(X + 1)(X^2 + 1).$$

And yet, for some prime  $p$ , in fact those congruent with 1 modulo 4, one finds 4 roots in  $\mathbb{Q}_p$ . For example, if  $p = 5$ ,  $f(X)$  factors as

$$\begin{aligned} f(X) = & (X - 1)(X - (4 + 4p + 4p^2 + 4p^3 + O(p^4))) \\ & \times (X - (2 + p + 2p^2 + p^3 + O(p^4)))(X - (3 + 3p + 2p^2 + 3p^3 + O(p^4))). \end{aligned}$$

One needs to know how to detect, from these 4 roots, which ones are possibly rational. In here, thanks to the periodic expansions of the first two roots, it is clear that they are the ones in  $\mathbb{Q}$ . One can notice that the other two roots are just representatives of  $i$  and  $-i$  in  $\mathbb{Q}_5$ .

In general, the period of a rational might be big enough not to be detected. One needs to know how to reconstruct a rational number from a  $p$ -adic expansion. Mignotte's bound [GG03, Chapter 6] is a bound on the size of the coefficients of the factors of such a polynomial  $f$ . In particular, this gives a bound on the precision needed to factor  $f$  in  $\mathbb{Q}_p$  in order to find the roots in  $\mathbb{Q}$ .

In the more general case of a solution  $x = (x_1, \dots, x_r) \in \mathbb{Q}_p^r$ , one may use the *rational reconstruction* (see [GG03, Section 5.10]) for each  $x_i$ , to see if they all are in  $\mathbb{Q}$ . If they are, then  $x \in \mathbb{Q}^r$  is a rational solution of the system. Bounds on the needed precision to do such a reconstruction exist, they are all linked to the so-called *arithmetic Bézout* (see [BGS94, KPS01, McK01]).

The following proposition gives us a complexity bound on the computation of a vector of recursive  $p$ -adic integers.

**Proposition 10.** *Let  $\Phi$  be an expression with  $L$  instructions of type addition, subtraction or product. Let  $\mathbf{y} \in \mathbb{Z}_p^r$  recursive of order  $k$  such that  $\mathbf{y} = \Phi(\mathbf{y})$  and  $\mathbf{y}_0, \dots, \mathbf{y}_k$  are known, then the computation of  $\mathbf{y}_n$  can be done in  $O(L \log_p(n) \log n)$  operations by making use of the relaxed product of Proposition 8.*

Among the different algorithms for algebraic systems solving, KRONECKER, presented in [GLS01, DL08], solves systems represented in evaluation, for example represented by a straight-line program (s.l.p.). The systems must admit only a finite number of solutions in the algebraic closure  $\bar{\mathbb{K}}$  of  $\mathbb{K}$ , they are said *zero-dimensional*. The main interest of such a representation is that some polynomials may have a lot of monomials but still be easily evaluated. For example  $(X_1 + \dots + X_r)^d$  can be evaluated in  $O(r + \log d)$  products thanks to the exponentiation by squaring, although, expanded, it has  $\binom{r+d-1}{d}$  monomials. Or the determinant of square matrix of size  $r$  which is a polynomial with  $r^2$  variables and  $r!$  terms but which can be evaluated in  $O(r^3)$  operations thanks to the Gaussian elimination.

In Chapter III, we give an automatic method for computing recursively a root  $y$  of a polynomial  $f(Y)$  such that  $y_0 = y \bmod p$  is a simple root of  $f(Y) \bmod p$ . First, let  $\Phi$  be function for which  $y$  is a fixed point, if and only if,  $y$  is a zero of  $f$ . In practice, we always take

$$\Phi(Y) = \frac{f'(y_0)Y - f(Y)}{f'(y_0)}.$$

It is not clear that such a function defines a recursive  $p$ -adic integer. We show that we can always find a function  $\Psi$  such that  $y$  is a recursive  $p$ -adic integer of order 1 and such that its evaluation complexity is linear in this of  $f$ . More precisely, we have the following proposition:

**Proposition 11. (Proposition III.18, [BL12, Proposition 18])** *Let  $f$  be a univariate polynomial over  $\mathbb{Z}_p$  given as a straight-line program such as its multiplicative complexity is  $L^*$ . Then there exists a function  $\Psi$  obtained from  $\Phi$  and  $y_0$  such that*

$$y = \Psi(y)$$

*and for all  $n \in \mathbb{N}^*$ , the computation  $\Psi(y)_n$  only needs  $y_0, \dots, y_{n-1}$ . Furthermore, the evaluation complexity of  $\Psi$  is bounded by  $2L^* + 1$ .*

We also show that the computation of the quotient of two  $p$ -adic integers can be extended to the computation of the solution of regular linear system. We conclude by dealing with the case of an algebraic system  $\mathbf{f}$  of  $r$  polynomial equations in  $r$  variables with coefficients in  $\mathbb{Z}$ . We assume that the reduction of  $\mathbf{f}$  modulo  $p$ ,  $\mathbf{f}_0$ , has a regular root  $\mathbf{y}_0$ , vector of  $p$ -adic integers. That is, we assume that the differential  $d\mathbf{f}_{\mathbf{y}_0}$  in  $\mathbf{y}_0$  is invertible over  $\mathbb{F}_p$ . This differential is then invertible over  $\mathbb{Z}_p$ , so that Hensel's lemma applies and Propositions 10 and 11 can be generalized to this situation.

**Proposition 12. (Proposition III.18, [BL12, Proposition 31])** *Let  $\mathbf{f}$  be a  $r$ -variate polynomial system over  $\mathbb{Z}_p$  given as a straight-line program such as its multiplicative complexity is  $L^*$ . Then there exists a function  $\Psi$  obtained from  $\Phi$  and  $\mathbf{y}_0$  such that*

$$\mathbf{y} = \Psi(\mathbf{y})$$

*and for all  $n \in \mathbb{N}^*$ , the computation  $\Psi(\mathbf{y})_n$  only needs  $\mathbf{y}_0, \dots, \mathbf{y}_{n-1}$ . Furthermore, the evaluation complexity of  $\Psi$  is bounded by  $3L^* + \frac{r(r+1)}{2}$ .*

In particular, we can hope to deduce an improvement in the complexity of KRONECKER algorithm by Giusti, Lecerf and Salvy (see [GLS01, DL08]).

We also do a complexity study when  $\mathbf{f}$  has dense representation, *i.e.* when each polynomial of  $\mathbf{f}$  is given by its vector of coefficients.

All the algorithms presented in Chapters II and III have been implemented in the C++ library ALGEBRAMIX for MATHEMAGIX [H+02]. Some examples of these implementations in C++ can be found in Appendix A. Examples for their use for computing the product, the quotient or the  $r$ th and  $p$ th roots with MATHEMAGIX are in Appendix B.

## d Reduction of bivariate polynomials

Chapter IV is a joint work with G. Lecerf entitled *Reduction of bivariate polynomials from convex-dense to dense, with application to factorizations* and accepted for publication in *Mathematics of Computation* [BL10]. We study the factorization of a bivariate polynomial. In particular, we look for reducing a polynomial  $f \in \mathbb{K}[X, Y]$  into a polynomial  $\tilde{f}$  whose factorization can be computed more easily in order to deduce this of  $f$ . Indeed, bivariate polynomial factorization is at the root of the decomposition of hypersurfaces into irreducible components, that is varieties defined by only one equation. Indeed, given a bivariate polynomial  $f(X, Y) \in \mathbb{K}[X, Y]$ , if one knows a factorization  $f(0, Y) = \tilde{f}_1(Y) \cdots \tilde{f}_s(Y)$  with  $\tilde{f}_1, \dots, \tilde{f}_s$  relatively prime, then by Hensel's lemma, there exist  $f_1(X, Y), \dots, f_s(X, Y) \in \mathbb{K}[[X]][Y]$  such that

$$f(X, Y) = f_1(X, Y) \cdots f_s(X, Y).$$

From a factor  $f_i(X, Y) = \sum_{j=0}^{d_i} f_{i,j}(X) Y^j$  one has to determine if the power series  $f_{i,j}(X) \in \mathbb{K}[[X]]$  are in  $\mathbb{K}(X)$ . One can use *Padé-Hermite approximants* [GG03, Section 5.9] and concludes that  $f$  factors in  $\mathbb{K}(X)[Y]$  and therefore in  $\mathbb{K}[X, Y]$ .

By repeating this process, one can deduce an algorithm for factoring  $f(X_1, \dots, X_r)$  in  $\mathbb{K}[X_1, \dots, X_r]$  as soon as  $f(0, \dots, 0, X_r)$  has coprime factors in  $\mathbb{K}[X_r]$ .

Assuming  $f$  is written

$$f(X, Y) = \sum_{(i,j) \in \mathbb{N}^2} f_{i,j} X^i Y^j,$$

the set  $\mathcal{S} = \{(i, j) \in \mathbb{N}^2, f_{i,j} \neq 0\}$  is called the *support* of  $f$ . The *Newton polygon*, denoted  $\mathcal{N}$ , is the convex hull of  $\mathcal{S}$ . Thanks to it and Newton-Puiseux algorithm, one can compute the roots of  $f$  when it is seen as a polynomial in  $Y$  with coefficients in  $\mathbb{K}[X]$  (see [Wal78, Chapter IV] and [Die68, Chapitre III, Appendice]). It helps also as an irreducible criterion for  $f$ , indeed, if  $f = gh$ , then  $\mathcal{N} = \mathcal{N}(g) + \mathcal{N}(h)$  [Ost21, Ost75, Ost99], where  $\mathcal{N}(g)$  and  $\mathcal{N}(h)$  are Newton polygons of  $g$  and  $h$  and  $+$  is the Minkowski sum. This criterion can be seen as a generalization of Eisenstein's criterion for bivariate polynomial.

The number of points with integral coordinates in  $\mathcal{N}$  is called the *convex size* of  $\mathcal{N}$  and is denoted  $\pi$ . The complexity bound for factoring is expected to depend on  $\pi$  or the area of  $\mathcal{N}$ , denoted  $\text{Vol } \mathcal{N}$ . For example, in [Wei10], under some conditions of genericity, one can find an algorithm for factoring  $f$  whose complexity bound is in  $O(\pi^\omega)$ , where  $\omega$  is the exponent of the linear algebra. And yet, the best complexity bounds for the *squarefree* and *irreducible factorizations* depend on the product of the partial degrees  $d_X$  in  $X$  and  $d_Y$  in  $Y$  (see for example [Gao03, Lec07, Lec08, Lec10]). Let us remark that, on the contrary, one cannot expect a complexity bound to depend on the cardinal  $\sigma$  of  $\mathcal{S}$ . Indeed, the polynomial  $f(X, Y) = X^p - Y^p \in \mathbb{Q}[X, Y]$ , for some prime  $p$ , is such that  $\sigma = 2$  but factors as

$$f(X, Y) = (X - Y)(X^{p-1} + X^{p-2}Y + \cdots + Y^{p-1}),$$

where the second factor has a support of size  $p$ .

Let us assume that  $d_X \geq d_Y$ . In characteristic zero, the squarefree factorization of  $f$  can be computed, in a deterministic way [Lec08, Proposition 8] (resp. probabilistic way [Lec08, Proposition 9]), in  $\tilde{O}(d_X d_Y^2)$  (resp.  $\tilde{O}(d_X d_Y)$ ) operations in  $\mathbb{K}$ . In positive characteristic  $p$ , the irreducible factorization of  $\mathbb{K}$  over  $\mathbb{K} = \mathbb{F}_{p^k}$  can be computed, in a probabilistic way [Lec10], in  $\tilde{O}(k (d_X d_Y)^{1.5})$  operations in  $\mathbb{F}_p$ .

These complexity bounds may seem rather pessimistic. For example, the polynomials family  $f_n(X, Y) = 1 + Y + X^n Y^n$  is such that the product of the partial degrees is  $n^2$  while the size of the Newton polygon  $\mathcal{N}_n$  is  $n/2$ .

The *bounding rectangle*  $\mathcal{R}$  of  $\mathcal{N}$  is the smallest rectangle of the form  $(o_X, o_Y) + [0, d_X] \times [0, d_Y]$  containing  $\mathcal{N}$ . The number of points with integral coordinates in  $\mathcal{R}$ , that is  $\delta = (d_X + 1)(d_Y + 1)$  is the *dense size* of  $\mathcal{N}$ .

**Theorem 13. (Theorems IV.2 and IV.20, [BL10, Theorems 1.2 and 4.3])**

Let  $\mathcal{S}$  be a finite subset of  $\mathbb{Z}^2$  of size  $\sigma$ , convex hull  $\mathcal{N}$ , convex size  $\pi$  and dense size  $\delta$ . Let  $\mathcal{R} = (o_X, o_Y) + [0, d_X] \times [0, d_Y]$  be the bounding rectangle of  $\mathcal{S}$ . There exists an invertible linear map  $U \in \text{Aff}(\mathbb{Z}^2)$  such that  $\tilde{\mathcal{N}} = U(\mathcal{N})$  is included in a rectangle  $\tilde{\mathcal{R}} = [0, \tilde{d}_X] \times [0, \tilde{d}_Y]$  verifying

$$\frac{3}{8} \tilde{d}_X \tilde{d}_Y \leq \text{Vol } \tilde{\mathcal{N}} \leq \tilde{d}_X \tilde{d}_Y. \quad (2)$$

Furthermore,  $U$  can be computed in  $O(\sigma \log^2 \pi)$  bit-operations and the dense size of  $\tilde{\mathcal{N}}$  is at most  $9\pi$ .

The map  $U$  can be computed as the composite function of *shear mappings*, axial symmetries and translations. From a geometric point of view on the curve defined by  $f$ ,  $U$  is the composition of blowups and blowdowns, so that the factorization of  $\tilde{f} = U(f)$  is essentially equivalent to that of  $f$ .

According to equation (2), one can notice that  $\text{Vol } \tilde{\mathcal{N}} \in O(\tilde{d}_X \tilde{d}_Y)$ . Though, as  $U$  is invertible over  $\mathbb{Z}$ , then  $\text{Vol } \tilde{\mathcal{N}} = \text{Vol } \mathcal{N}$  and  $\pi \in O(\tilde{d}_X \tilde{d}_Y)$ . One deduces that by reducing  $f$  to  $\tilde{f}$ , computing the wanted factorization of  $\tilde{f}$  and applying  $U^{-1}$  to its factors, one can find the squarefree factorization of  $f$ , in a deterministic (resp. probabilistic) way, in  $\tilde{O}(\pi^{1.5})$  (resp.  $\tilde{O}(\pi)$ ) operations in  $\mathbb{K}$  when  $\text{char } \mathbb{K} = 0$  and the irreducible factorization, in a probabilistic way, of  $f$  in  $\tilde{O}(k \pi^{1.5})$  operations in  $\mathbb{F}_p$  when  $\mathbb{K} = \mathbb{F}_{p^k}$ .

At last, we also show that the  $3/8$  factor appearing in equation (2) is optimal, in general.

**Proposition 14. (Proposition IV.21, [BL10, Proposition 4.4])** Let  $\mathcal{S}$  be a subset of  $\mathbb{Z}^2$ . With the convention that  $\frac{\text{Vol } U(\mathcal{S})}{\text{Vol } \mathcal{R}(U(\mathcal{S}))} = 1$  whenever  $\text{Vol } \mathcal{S} = 0$ , one has

$$\inf_{\mathcal{S} \subset \mathbb{Z}^2, |\mathcal{S}| < \infty} \sup_{U \in \text{Aff}(\mathbb{Z}^2)} \frac{\text{Vol } U(\mathcal{S})}{\text{Vol } \mathcal{R}(U(\mathcal{S}))} = \frac{3}{8},$$

where  $\mathcal{R}(U(S))$  represents the bounding rectangle of  $U(S)$ .

## e Resolution in average of real algebraic systems

In Chapter V, we are interested, with L. M. Pardo, in the computation in average of an approximate root of a real algebraic system. This chapter is based on an article named *Spherical Radon transform and the average of the condition number on certain Schubert subvarieties of a Grassmannian* accepted for publication in *Journal of Complexity* [BP11b]

In [SS93a, SS93b, BCSS98], the authors define an approximate zero  $z'$  of an actual zero  $z$  of a homogeneous complex algebraic system  $f = (f_1, \dots, f_n) \in \mathbb{C}[X_0, \dots, X_n]^n$ , as an element of  $\mathbb{P}^n(\mathbb{C})$  for which the iteration of Newton operator converges immediately quadratically to  $z$ . Let  $d_R$  be the natural Riemannian distance over the unit-sphere of  $\mathbb{C}^{n+1}$  and  $d_{\mathbb{P}}$  be projective distance over  $\mathbb{P}^n(\mathbb{C})$  defined by  $d_{\mathbb{P}}(w, w') = \sin(d_R(w, w'))$ . Formally speaking, denote  $z_0 = z'$  and for all  $k \in \mathbb{N}$ ,  $z_{k+1} = z_k - d_{f_z|_{T_{f(z)}}^{-1}}(f(z))$ , then for all  $k$ , one has

$$d_{\mathbb{P}}(z, z_k) \leq \left(\frac{1}{2}\right)^{2^k - 1} d_{\mathbb{P}}(z, z').$$

Smale's seventeenth problem is about whether or not there exists an algorithm for computing in polynomial time an approximate zero  $z'$  of a homogeneous generic system  $f$  of  $n$  polynomials in  $n + 1$  variables. The algorithm presented by Beltrán and Pardo [BP09a, BP09b, BP11a] is an example of such an algorithm whose complexity bound is in  $O(N^2)$ , where  $N$  is the dense size of  $f$ . However, their algorithm is probabilistic. Today, the best known deterministic algorithm is an adaptation of theirs thanks to the *smooth analysis*, the exponent appearing in its complexity is in  $O(\log \log N)$ .

Let us denote  $d_i = \deg f_i$  and  $(d) = (d_1, \dots, d_n)$ . Let us denote also  $\mathcal{H}_{(d)}$  the space of complex homogeneous systems  $g = (g_1, \dots, g_n)$  in variables  $X_0, \dots, X_n$  such that  $\deg g_i = d_i$ . This space is naturally endowed with Bombieri's norm and one can assume that each system in the following has norm 1, in particular,  $f$  is assumed to have norm 1. Beltrán and Pardo propose to apply homotopic deformation technique along a great circle of  $\mathbb{S}(\mathcal{H}_{(d)})$ , the unit-sphere of  $\mathcal{H}_{(d)}$ . From a system  $g$  for which they know a root  $\zeta$ , they study how  $\zeta$  evolves along the great circle passing through  $g$  and  $f$ . For Beltrán and Pardo's algorithm to return an approximate root of  $f$ , the great circle must not pass through  $\Sigma$ , the discriminant variety. This variety  $\Sigma$  is the variety of the *ill-conditioned* systems of  $\mathbb{S}(\mathcal{H}_{(d)})$ , that is of the systems whose set of zeros has not dimension zero or whose at least one zero is multiple. This is a variety of complex codimension 1 and therefore of real codimension 2.

The authors introduced the *questor set*  $\mathcal{G}_{(d)}$  which is a set of systems  $g$  for which they know how to compute an exact root  $\zeta$ . By picking at random a pair  $(g, \zeta)$  in  $\mathcal{G}_{(d)}$ , with probability 1, the great circle passing through  $g$  and  $f$  does not intersect  $\Sigma$ . Therefore, with probability 1, Beltrán and Pardo's algorithm returns an approximate root of  $f$ .

In Chapter V, we deal with the analogous case wherein  $f$  is assumed to have real coefficients, which we denote  $f \in \mathcal{H}_{(d)}^{\mathbb{R}}$ . A straightforward adaptation of Beltrán and Pardo's algorithm is not possible. Indeed,  $\Sigma^{\mathbb{R}} = \Sigma \cap \mathbb{S}(\mathcal{H}_{(d)}^{\mathbb{R}})$ , the real discriminant variety, cuts the unit-sphere  $\mathbb{S}(\mathcal{H}_{(d)}^{\mathbb{R}})$  off in an exponential number of connected components. Therefore if we want to apply homotopic deformation technique, we have to pick up  $g$  in the same component as  $f$ , which is not reasonable. That is why, we propose to choose  $g$  among the complex systems instead of only the real systems and then to use their algorithm. We deduce that our variant of Beltrán and Pardo's algorithm returns an approximate root of  $f$  with probability 1.

Let us denote  $\mu_{\text{norm}}(g, \zeta)$  the normalized condition number of  $g$  at its root  $\zeta$ . This number determine how much a small perturbation of  $g$  affects  $\zeta$ . It is linked to the condition number of the Jacobian matrix of  $g$  at  $\zeta$  and therefore, in particular, to the operator norm of the inverse of  $d g_{\zeta}$  which can be denoted  $\|d g_{\zeta}^{-1}\|$ . The number of homotopic deformation steps performed by the Newton operator starting with a pair  $(g, \zeta)$  and targeting  $f$  along a great circle  $L$  passing through  $g$  and  $f$  is bounded by

$$\mathcal{C}(f, g, \zeta) = \int_{h \in L} \mu_{\text{norm}}(h, \xi)^2 d L,$$

where  $\xi$  is the zero of  $h$  obtained from the perturbation of  $\zeta$  (see [Shu09]). A complexity bound of our algorithm is given by the expectation  $E$  defined by

$$E = E_{\mathbb{S}(\mathcal{H}_{(d)}^{\mathbb{R}}) \times \mathbb{S}(\mathcal{G}_{(d)})}(\mathcal{C}(f, g, \zeta)).$$

In the computation of this expectation, one can use the average of the square of the condition numbers  $\mu_{\text{av}}^2$  defined as follows:

$$\mu_{\text{av}}^2(g) = \frac{1}{|\mathbb{V}(g)|} \sum_{\zeta \in \mathbb{V}(g)} \mu_{\text{norm}}(g, \zeta)^2.$$

Let us denote  $N + 1 = \dim_{\mathbb{C}} \mathcal{H}_{(d)} = \dim_{\mathbb{R}} \mathcal{H}_{(d)}^{\mathbb{R}}$ , therefore,  $N$  is the dense size of a system in  $\mathbb{S}(\mathcal{H}_{(d)}^{\mathbb{R}})$ . Thanks to the spherical Radon transform [Rub02] defined by

$$\mathbf{R}^{\alpha}[\mu_{\text{av}}^2](\mathbb{S}(\mathcal{H}_{(d)}^{\mathbb{R}})) = \frac{\mathbf{B}\left(\frac{N-\alpha+2}{2}, \frac{\alpha+N+1}{2}\right)}{\text{vol } \mathbb{S}(\mathcal{H}_{(d)})} \int_{\mathbb{S}(\mathcal{H}_{(d)})} \frac{\mu_{\text{av}}^2(g)}{d_{\mathbb{P}}(g, \mathbb{S}(\mathcal{H}_{(d)}^{\mathbb{R}}))^{N+1-\alpha}} d \mathbb{S}(\mathcal{H}_{(d)}),$$

we deduce the following result:

**Theorem 15.** (Corollary V.13, [BP11b, Corollary 12]) *Let us assume  $\dim_{\mathbb{R}} \mathcal{H}_{(d)}^{\mathbb{R}} = N + 1$  and denote*

$$C(2N + 1, N, i) = 2 \binom{N-1}{2} \frac{\mathbf{B}\left(\frac{2N+3}{2}, \frac{1}{2}\right)}{\mathbf{B}\left(\frac{N-1}{2}, \frac{1}{2}\right)}.$$

*Then the bound  $E$  of the complexity of our real variant of Beltrán and Pardo's algorithm verifies,*

1. *if  $N + 1$  is even,*

$$E = \sum_{i=0}^{\frac{N-1}{2}} C(2N + 1, N, i) \mathbf{R}^{N-2i}[\mu_{\text{av}}^2](\mathbb{S}(\mathcal{H}_{(d)}^{\mathbb{R}}));$$

2. if  $N + 1$  is odd,

$$\begin{aligned} \mathbf{R}^1[\mu_{\text{av}}^2](\mathbb{S}(\mathcal{H}_{(d)}^{\mathbb{R}})) &\leq E \leq \sqrt{2} \left( \frac{\mathbf{R}^1[\mu_{\text{av}}^2](\mathbb{S}(\mathcal{H}_{(d)}^{\mathbb{R}}))}{\mathbf{B}(\frac{N+1}{2}, \frac{N}{2})} \right) \\ E &\leq \sqrt{2} E_{\mathbb{S}(\mathcal{H}_{(d)})} \left[ \frac{\mu_{\text{av}}^2(g)}{\mathbf{d}_{\mathbb{P}}(g, \mathbb{S}(\mathcal{H}_{(d)}^{\mathbb{R}}))^N} \right]. \end{aligned}$$

In fact, our results are a little more general. Given a  $\mathbb{R}$ -vector subspace  $M$  of  $\mathcal{H}_{(d)}$ , we compute the expectation  $E = E_{\mathbb{S}(M) \times \mathbb{S}(\mathcal{G}_{(d)})}(\mathcal{C}(f, g, \zeta))$  for finding an approximate root of  $f \in M$  with norm 1 starting with a pair  $(g, \zeta)$  in  $\mathcal{G}_{(d)}$ . Above, the space  $M$  was this of real systems  $\mathcal{H}_{(d)}^{\mathbb{R}}$ , but it can be the space of algebraic systems with support included in a given set  $\mathcal{S}$ . Theorem 15 is then a special case of the following theorem:

**Theorem 16. (Corollary V.10, [BP11b, Corollary 9])** *Let  $N + 1 = \dim_{\mathbb{C}} \mathcal{H}_{(d)}$  and  $p + 1 = \dim_{\mathbb{R}} M$ . For all  $i$ , denote*

$$\begin{aligned} B_0(2N + 1, p, i) &= 2 \binom{N - 1 - \frac{p}{2}}{2} \frac{\mathbf{B}(N + \frac{3}{2}, \frac{1}{2})}{\mathbf{B}(\frac{p-1}{2}, \frac{1}{2})}, \\ C(2N + 1, p, i) &= 2 \binom{N - \frac{p+1}{2}}{i} \frac{\mathbf{B}(N + \frac{3}{2}, \frac{1}{2})}{\mathbf{B}(\frac{p-1}{2}, \frac{1}{2})}. \end{aligned}$$

Let  $E$  be the complexity bound for the variant of Beltrán and Pardo's algorithm,

1. if  $\text{codim}_{\mathbb{R}} M = 1$ , then

$$\frac{4\sqrt{2\pi}}{(2N + 1 + \sqrt{3})^{1/2}} E_{\mathbb{S}(\mathcal{H}_{(d)})}[\mu_{\text{av}}^2] \leq E \leq \sqrt{\frac{(2N - 1)\pi}{2}} \mathbf{R}^0[\mu_{\text{av}}^2](\mathbb{S}(M));$$

2. if  $\text{codim}_{\mathbb{R}} M$  is even, then

$$E = \sum_{i=0}^{\frac{2N-p-1}{2}} C(2N + 1, p, i) \mathbf{R}^{2(N-i)-p}[\mu_{\text{av}}^2](\mathbb{S}(M));$$

3. if  $\text{codim}_{\mathbb{R}} M$  is odd and greater than 1, then  $E$  is bounded by the following quantities:

$$\begin{aligned} E &\geq \sum_{i=0}^{\frac{2N-p-2}{2}} \frac{(2N - 1) B_0(2N + 1, p, i)}{\sqrt{i + \sqrt{3/2}}} \mathbf{R}^{2(N-i)-p-1}[\mu_{\text{av}}^2](\mathbb{S}(M)), \\ E &\leq \sum_{i=0}^{\frac{2N-p-2}{2}} \frac{8 B_0(2N + 1, p, i)}{(2i + 1)(2N - 1)} \mathbf{R}^{2(N-i)-p-1}[\mu_{\text{av}}^2](\mathbb{S}(M)). \end{aligned}$$

## Bibliography

[BCSS98] L. Blum, F. Cucker, M. Shub and S. Smale. *Complexity and real computation*. Springer-Verlag, New York, 1998. With a foreword by Richard M. Karp.

- [BGS94] J.-B. Bost, H. Gillet and C. Soulé. Heights of projective varieties and positive Green forms. *J. Amer. Math. Soc.*, 7(4):903–1027, 1994.
- [BHL11] J. Berthomieu, J. van der Hoeven and G. Lecerf. Relaxed algorithms for  $p$ -adic numbers. *J. Théor. Nombres Bordeaux*, 23(3):541–577, 2011.
- [BHM10] J. Berthomieu, P. Hivert and H. Mourtada. Computing Hironaka’s invariants: Ridge and Directrix. In *Arithmetic, Geometry, Cryptography and Coding Theory 2009*, volume 521 of *Contemp. Math.*, pages 9–20. Amer. Math. Soc., Providence, RI, 2010.
- [BL10] J. Berthomieu and G. Lecerf. Reduction of bivariate polynomials from convex-dense to dense, with application to factorization. Manuscript to appear in *Math. Comp.*, 2010.
- [BL12] J. Berthomieu and R. Lebreton. Relaxed  $p$ -adic Hensel lifting for algebraic systems. Work in progress, 2011.
- [BP09a] C. Beltrán and L. M. Pardo. Efficient polynomial system-solving by numerical methods. *J. Fixed Point Theory Appl.*, 6(1):63–85, 2009.
- [BP09b] C. Beltrán and L. M. Pardo. Smale’s 17th problem: Average polynomial time to compute affine and projective solutions. *J. Amer. Math. Soc.*, 22(2):363–385, 2009.
- [BP11a] C. Beltrán and L. M. Pardo. Fast linear homotopy to find approximate zeros of polynomial systems. *Found. Comput. Math.*, 11(1):95–129, 2011.
- [BP11b] J. Berthomieu and L. M. Pardo. Spherical Radon transform and the average of the condition number on certain Schubert subvarieties of a Grassmannian. Manuscript to appear in *J. Complexity*, 2011.
- [Coh46] I. S. Cohen. On the structure and ideal theory of complete local rings. *Trans. Amer. Math. Soc.*, 59:54–106, 1946.
- [CP08] V. Cossart and O. Piltant. Resolution of singularities of threefolds in positive characteristic. I. Reduction to local uniformization on Artin-Schreier and purely inseparable coverings. *J. Algebra*, 320(3):1051–1082, 2008.
- [CP09] V. Cossart and O. Piltant. Resolution of singularities of threefolds in positive characteristic. II. *J. Algebra*, 321(7):1836–1976, 2009.
- [Die68] J. Dieudonné. *Calcul infinitésimal*. Hermann, Paris, 1968.
- [DL08] C. Durvye and G. Lecerf. A concise proof of the Kronecker polynomial system solver from scratch. *Expositiones Mathematicae*, 26(2):101–139, 2008.
- [FS56] A. Fröhlich and J. C. Shepherdson. Effective procedures in field theory. *Philos. Trans. Roy. Soc. London. Ser. A.*, 248:407–432, 1956.
- [Gao03] S. Gao. Factoring multivariate polynomials via partial differential equations. *Math. Comp.*, 72:801–822, 2003.
- [GG03] J. von zur Gathen and J. Gerhard. *Modern computer algebra*. Cambridge University Press, Cambridge, Second edition, 2003.
- [Gir72] J. Giraud. *Étude locale des singularités*. U.E.R. Mathématique, Université Paris XI, Orsay, 1972. Cours de 3ème cycle, 1971–1972, Publications Mathématiques d’Orsay, No. 26.
- [Gir75] J. Giraud. Contact maximal en caractéristique positive. *Ann. Sci. École Norm. Sup. (4)*, 8(2):201–234, 1975.
- [GLS01] M. Giusti, G. Lecerf and B. Salvy. A Gröbner free alternative for polynomial system solving. *J. Complexity*, 17(1):154–211, 2001.
- [G+91] T. Granlund et al. GMP, the GNU multiple precision arithmetic library. 1991. Available from <http://www.swox.com/gmp>.
- [Hal01] E. Hallouin. Computing local integral closures. *J. Symbolic Comput.*, 32(3):211–230, 2001.

- [Hir64] H. Hironaka. Resolution of singularities of an algebraic variety over a field of characteristic zero. I, II. *Ann. of Math. (2)* 79 (1964), 109–203; *ibid. (2)*, 79:205–326, 1964.
- [Hir67] H. Hironaka. Characteristic polyhedra of singularities. *J. Math. Kyoto Univ.*, 7:251–293, 1967.
- [Hir70] H. Hironaka. Additive groups associated with points of a projective space. *Ann. of Math. (2)*, 92:327–334, 1970.
- [H+02] J. van der Hoeven et al. Mathemagix. 2002. Available from <http://www.mathemagix.org>.
- [Hoe94] M. van Hoeij. An algorithm for computing an integral basis in an algebraic function field. *J. Symbolic Comput.*, 18(4):353–363, 1994.
- [Hoe97] J. van der Hoeven. Lazy multiplication of formal power series. In W. W. K uchlin, editor, *ISSAC '97: Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation*, pages 17–20. Maui, Hawaii, 1997.
- [Hoe02] J. van der Hoeven. Relax, but don't be too lazy. *J. Symbolic Comput.*, 34(6):479–542, 2002.
- [KPS01] T. Krick, L. M. Pardo and M. Sombra. Sharp estimates for the arithmetic Nullstellensatz. *Duke Math. J.*, 109(3):521–598, 2001.
- [Lan02] S. Lang. *Algebra*, volume 211 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, third edition, 2002.
- [Lec07] G. Lecerf. Improved dense multivariate polynomial factorization algorithms. *J. Symbolic Comput.*, 42(4):477–494, 2007.
- [Lec08] G. Lecerf. Fast separable factorization and applications. *Appl. Algebr. Engrg. Comm. Comput.*, 19(2):135–160, 2008.
- [Lec10] G. Lecerf. New recombination algorithms for bivariate polynomial factorization based on Hensel lifting. *Appl. Algebr. Engrg. Comm. Comput.*, 21(2):151–176, 2010.
- [McK01] D. McKinnon. An arithmetic analogue of B ezout's theorem. *Compositio Math.*, 126(2):147–155, 2001.
- [Ost21] A. M. Ostrowski.  ber die Bedeutung der Theorie der konvexen Polyeder f ur die formale Algebra. *Jahresber. Deutsch. Math.-Verein.*, 30(2):98–99, 1921. Talk given at *Der Deutsche Mathematikertag vom 18–24 September 1921 in Jena*.
- [Ost75] A. M. Ostrowski. On multiplication and factorization of polynomials. I. Lexicographic orderings and extreme aggregates of terms. *Aequationes Math.*, 13(3):201–228, 1975.
- [Ost99] A. M. Ostrowski. On the significance of the theory of convex polyhedra for formal algebra. *SIGSAM Bull.*, 33(1):5, 1999. Translated from [Ost21].
- [Rub02] B. Rubin. Inversion formulas for the spherical Radon transform and the generalized cosine transform. *Adv. in Appl. Math.*, 29(3):471–497, 2002.
- [Sha94] I. R. Shafarevich. *Basic algebraic geometry. 1*. Springer-Verlag, Berlin, Second edition, 1994. Varieties in projective space, Translated from the 1988 Russian edition and with notes by Miles Reid.
- [Shu09] M. Shub. Complexity of B ezout's theorem. VI. Geodesics in the condition (number) metric. *Found. Comput. Math.*, 9(2):171–178, 2009.
- [SS93a] M. Shub and S. Smale. Complexity of B ezout's theorem. I. Geometric aspects. *J. Amer. Math. Soc.*, 6(2):459–501, 1993.
- [SS93b] M. Shub and S. Smale. Complexity of B ezout's theorem. II. Volumes and probabilities. In *Computational algebraic geometry (Nice, 1992)*, volume 109 of *Progr. Math.*, pages 267–285. Birkh user Boston, Boston, MA, 1993.
- [Tra84] B. M. Trager. *Integration of algebraic functions*. PhD thesis, Massachusetts Institute of Technology, 1984.

- [**Wa178**] R. J. Walker. *Algebraic curves*. Springer-Verlag, New York, 1978. Reprint of the 1950 edition.
- [**Wei10**] M. Weimann. A lifting and recombination algorithm for rational factorization of sparse polynomials. *J. Complexity*, 26(6):608–628, 2010.

# Annexe A

## Implementations in C++

In this appendix, we give some examples of our implementation in C++ for MATH-EMAGIX of algorithms for  $p$ -adic integers. In the following, we present the source code of a  $p$ -adic integer obtained from the application of a binary operator  $\text{Op}$  to other  $p$ -adics  $f$  and  $g$ , for example, operator  $\text{Op}$  can be the binary plus or minus. The operator  $\text{Op}$  has to apply coefficient-wise on  $f$  and  $g$  with maybe a propagation of a carry. We assume furthermore that  $\mathbb{C}$  is the ground ring and that  $\mathbb{M}$  is a system of representatives of  $\mathbb{C}/(p)$  such as  $\{0, \dots, p-1\}$  if  $\mathbb{C} = \mathbb{Z}$ . At last,  $V$  is a variant, in our case, it is derived either from `series_carry_naive` when ones needs to use `series` for which the operations may involved a carry or from `series_naive` when one needs to use classical formal power series.

```
#define Series series<M, V>
#define Series_rep series_rep<M, V>
typedef unsigned int nat;

template<typename Op, typename M, typename V>
class binary_series_rep: public Series_rep {
protected:
    const Series f, g;
    C carry;
public:
    inline binary_series_rep (const Series& f2, const Series& g2):
        f (f2), g (g2), carry (0) {}
    virtual void Increase_order (nat l) {
        Series_rep::Increase_order (l);
        increase_order (f, l);
        increase_order (g, l);
    }
    virtual M next () {
        return Op::op_mod (f[this->n].rep, g[this->n].rep,
            M::get_modulus (), carry);
    }
};
```

In this piece of code, `Op::op_mod` in the `next ()` function applies the operator  $\text{Op}$  to the  $n$ th coefficients of  $f$  and of  $g$ , adds the possible carry and then take the remainder modulo  $p$ . Finally, the quotient is stored in `carry`.

The method `increase_order (nat l)` is called when one wants to know the current series up till precision  $l$ . It takes care of increasing the precision of the different  $p$ -adics involved in the computation of said  $p$ -adic.

The following piece of code represents our implementation of the relaxed multiplication with conversions between base  $p$  and base 2 as described in Chapter II Section 3.2.

```

template<typename M, typename V>
class mul_series_rep: public Series_rep {
public:
    typedef Lift_type (M) L;

protected:
    Series f, g;
    int lnz_f; // last non-zero coefficient in f
    int lnz_g; // last non-zero coefficient in g
    nat zeros_f; // bit  $2^p$  set if  $f[2^p-1 \dots 2^{p+1}-1] = 0$ 
    nat zeros_g; // bit  $2^p$  set if  $g[2^p-1 \dots 2^{p+1}-1] = 0$ 

    vector<L> b_f, d_f, b_g, d_g; // binary representations for the segments
    // b_f[q] is the last segment of f of size  $2^q$ 
    // d_f[q] is the "diagonal segment" of size  $2^q$ 
    vector<L> powers_of_p; //  $p^{(2^i)}$ 
    vector<L> carry; // carry[q] is the carry in size  $2^q$ 

L get_power_of_p (const Modulus& p, nat i) {
    // lazy access to  $p^{(2^i)}$ , for  $i \geq 0$ 
    // with its own memory allocation
    static const L zero (0);
    while (i >= N (powers_of_p))
        powers_of_p << vector<L> (zero, max ((nat) 1, N (powers_of_p)));
    if (powers_of_p [i] == zero) {
        if (i == 0) powers_of_p[0] = L(* p);
        else powers_of_p[i] = square (get_power_of_p (p, i - 1));
    }
    return powers_of_p [i];
}

public:
    mul_series_rep (const Series& f2, const Series& g2):
        f (f2), g (g2), lnz_f (-1), lnz_g (-1), zeros_f (0), zeros_g (0) {}
    ~mul_series_rep () {}

    void Set_order (nat l2) {
        Series_rep::Set_order (l2);
        nat m = log_2 (next_power_of_two (l2 + 1));
        if (N(b_f) < m) {
            vector<L> tmp (L(0), m - N(b_f));
            b_f << tmp; d_f << tmp;
            b_g << tmp; d_g << tmp;
            carry << tmp;
        }
    }

    void Increase_order (nat l) {
        Set_order (l);
        increase_order (f, l);
        increase_order (g, l);
    }

M next () {
    Modulus p = M::get_modulus ();
    nat k = this->n;
    if (exact_neq (f[k], M(0))) lnz_f = k;
    if (exact_neq (g[k], M(0))) lnz_g = k;
    if (k == 0) {
        C r = 0;
        if (lnz_f < 0) zeros_f |= 1;
        if (lnz_g < 0) zeros_g |= 1;
    }
}

```

```

    if ((zeros_f&1) == 0 && (zeros_g&1) == 0) {
        C q= 0;
        mul_mod (r, * f[0], * g[0], p, q);
        carry[0]= as<L> (q);
    }
    b_f[0]= d_f[0]= lift (f[0]);
    b_g[0]= d_g[0]= lift (g[0]);
    return M (r, true);
}
else {
    k = 2 * (this->n + 2);
    L t= 0, e_f= lift (f[this->n]), e_g= lift (g[this->n]);
    nat q= -1;
    while ((k&1) == 0 && k != 2) {
        q++; k= k >> 1;
        if (q > 0) {
            e_f= b_f[q-1] + get_power_of_p (p, q-1) * e_f;
            e_g= b_g[q-1] + get_power_of_p (p, q-1) * e_g;
        }
        if (k == 2) {
            if (lnz_f < ((int) (1<<q)-1) zeros_f |= (1<<q);
            if (lnz_g < ((int) (1<<q)-1) zeros_g |= (1<<q);
            d_f[q]= e_f; d_g[q]= e_g;
        }
        t += carry[q];
        if (q == 0 ||
            ((zeros_f & (1<<q)) == 0 && lnz_g >= ((int) (((k-1)<<q)-1))))
            t += d_f[q] * e_g;
        if (k == 2) break;
        if (q == 0 ||
            ((zeros_g & (1<<q)) == 0 && lnz_f >= ((int) (((k-1)<<q)-1)))) {
            t += e_f * d_g[q];
        }
    }
    b_f[q]= e_f; b_g[q]= e_g;
    while (q != (nat) -1) {
        t = rem (t, get_power_of_p (p, q), carry[q]);
        q--;
    }
    return M (as<C> (t), true);
}
};

```

Finally, this last class is a recursive  $p$ -adic integer as presented in Chapter II Section 5 and in particular, the quotient of two  $p$ -adic integers as in Section 6.2.

```

#define Recursive_series_rep recursive_series_rep<M, V>
template<typename M, typename V>
class div_series_rep: public Recursive_series_rep {
protected:
    Series f, g;
public:
    div_series_rep (const Series& f2, const Series& g2):
        f(f2), g(g2) {}
    virtual void Increase_order (nat l) {
        Recursive_series_rep::Increase_order (l);
        increase_order (f, l);
        increase_order (g, l);
    }
    Series initialize () {

```

```

typedef typename Series_variant(C) SV;
ASSERT (g[0] != 0, "division by zero");
M u= invert (g[0]);
this->initial (0)= u * f[0];
Series tmp= f - lshiftz (this -> me () * rshiftz (g))
- carry_mul_series (this -> me (), g[0]);
return as<Series> (u * as<series<M,SV> > (tmp));
}
};

```

Method `initialize ()` takes care of assigning the first coefficients of the current series, with `this->initial (nat k)` and returns an equation for which the series is a fixed point.

In this equation, `lshiftz (Series s)` (resp. `rshiftz (Series s)`) is just a division (resp. multiplication) by  $p$ , `carry_mul_series (Series a, M b)` is our implementation of `mul_quo` and finally, `this->me ()` returns said series at its current stage of computation so that further coefficients may be computed. In the last line, `as<series<M,SV> >` converts the  $p$ -adic integer `tmp` into a series over  $\mathbb{F}_p$ , so that the next product behaves like `mul_rem`. In the end, the obtained series is converted back into a  $p$ -adic integer. The returned expression is used in the `next ()` method defined in `recursive_series_rep`, the method evaluates the expression at the current stage and returns its  $n$ th coefficient.

# Annexe B

## Examples in MATHEMAGIX

In this appendix, we present how to use our implementations of  $p$ -adic integers, presented in Chapter II, in MATHEMAGIX, in particular with the interpreter MATHEMAGIX-LIGHT.

```
Welcome to Mathemagix-light 0.4
This software falls under the GNU General Public License
It comes without any warranty whatsoever
www.mathemagix.org
(c) 2001-2010
```

### a The basics with $p$ -adic integers

First of all, we tell MATHEMAGIX that we need to use the library ALGEBRAMIX.

```
Mmx] use "algebramix"
```

Let us create two 13-adic integers  $x$  and  $y$  such that  $x \in \mathbb{N}$  and  $y \in \mathbb{Z}$ .

In the following, `@p_expansion (a, modulus p)` converts a natural integer  $a$  into a vector of its coefficients in base  $p$ . Then, `p_adic (v)` creates a  $p$ -adic integer from this vector:

```
Mmx] x: P_adic Modular Integer == p_adic (@p_expansion (1234567890,
modulus 13));
mmout << "x = " << x << ".";
```

$$x = 10 + 5p + 6p^2 + 8p^3 + 10p^5 + 8p^6 + 6p^7 + p^8 + O(p^{10}).$$

```
Mmx] y: P_adic Modular Integer == -p_adic(@p_expansion (9876543210,
modulus 13));
mmout << "y = " << y << ".";
```

$$y = 4 + 3p + 4p^2 + 3p^3 + 10p^4 + 5p^5 + 6p^6 + 5p^7 + 11p^8 + 12p^9 + O(p^{10}).$$

By default, the precision until which a  $p$ -adic integer is printed is 10. Let us increase the precision up till 20 and print  $x$  and  $y$ . This is done thanks to the `set_output_order (x, 1)` function which takes as its first argument any `P_adic Modular Integer` and as its second argument the precision.

```
Mmx] set_output_order (x, 20);
mmout << "x = " << x << ",\n\ny = " << y << ".";
```

$$x = 10 + 5p + 6p^2 + 8p^3 + 10p^5 + 8p^6 + 6p^7 + p^8 + O(p^{20}),$$

$$y = 4 + 3p + 4p^2 + 3p^3 + 10p^4 + 5p^5 + 6p^6 + 5p^7 + 11p^8 + 12p^9 + 12p^{10} + 12p^{11} + 12p^{12} + 12p^{13} + 12p^{14} + 12p^{15} + 12p^{16} + 12p^{17} + 12p^{18} + 12p^{19} + O(p^{20}).$$

Basic ring operations are done in a classic way using +, -, \*.

```
Mmx] mmout << "x + y = " << (x+y) << ", \n\nx - y = " << (x-y)
      << ", \n\nx * y = " << (x*y) << ".";
```

$$x + y = 1 + 9p + 10p^2 + 11p^3 + 10p^4 + 2p^5 + 2p^6 + 12p^7 + 12p^8 + 12p^9 + 12p^{10} + 12p^{11} + 12p^{12} + 12p^{13} + 12p^{14} + 12p^{15} + 12p^{16} + 12p^{17} + 12p^{18} + 12p^{19} + O(p^{20}),$$

$$x - y = 6 + 2p + 2p^2 + 5p^3 + 3p^4 + 4p^5 + 2p^6 + p^7 + 3p^8 + O(p^{20}),$$

$$x * y = 1 + p + 5p^2 + 2p^3 + 2p^4 + 8p^5 + 12p^6 + 10p^7 + 5p^8 + 12p^9 + 2p^{10} + 6p^{11} + 7p^{12} + 7p^{13} + 12p^{14} + 7p^{15} + 10p^{16} + 12p^{17} + 12p^{18} + 12p^{19} + O(p^{20}).$$

Likewise, the division is performed by using /, when possible, *i.e.* when the divisor is not in  $p\mathbb{Z}_p$ .

```
Mmx] set_output_order (x, 50); z == x/y;
      mmout << "z = x/y = " << z << ".";
```

$$z = x/y = 9 + 7p + 9p^2 + 9p^3 + 3p^4 + 4p^5 + 12p^6 + 8p^8 + 10p^9 + 8p^{10} + 4p^{11} + p^{12} + 11p^{13} + 2p^{14} + 12p^{15} + 12p^{16} + p^{17} + 7p^{18} + 7p^{19} + 11p^{20} + 5p^{21} + 12p^{22} + 9p^{23} + 11p^{24} + 7p^{25} + 10p^{26} + 12p^{27} + 5p^{28} + 5p^{30} + 2p^{31} + 10p^{32} + 10p^{33} + 8p^{34} + 9p^{35} + 2p^{37} + 4p^{38} + 9p^{39} + 3p^{40} + 7p^{41} + 7p^{42} + 8p^{43} + 9p^{44} + 5p^{45} + p^{46} + 9p^{47} + 8p^{48} + 9p^{49} + O(p^{50}).$$

The multiplication by  $p^r$  can either be performed by creating a  $p$ -adic integer equals to  $p$  and then by multiplying by this  $p$ -adic  $r$  times:

```
Mmx] p: P_adic Modular Integer == p_adic (@p_expansion
      (13, modulus 13));
      mmout << "p = " << p << ", \n\nx*p^3 = " << x*p*p*p << ", ";
```

$$p = p + O(p^{50}),$$

$$x*p^3 = 10p^3 + 5p^4 + 6p^5 + 8p^6 + 10p^8 + 8p^9 + 6p^{10} + p^{11} + O(p^{50}),$$

or it can be performed by shifting by  $r$  with the operator << r:

```
Mmx] mmout << "x*p^3 = " << (x << 3) << ".";
```

$$x*p^3 = 10p^3 + 5p^4 + 6p^5 + 8p^6 + 10p^8 + 8p^9 + 6p^{10} + p^{11} + O(p^{50}).$$

The reciprocal operator >> r is a shift in the other way, it returns the quotient of a division by  $p^r$ :

```
Mmx] mmout << "x/p^2 = " << (x >> 2) << ".";
```

$$x/p^2 = 6 + 8p + 10p^3 + 8p^4 + 6p^5 + p^6 + O(p^{50}).$$

## b Computation of $r$ th roots in $\mathbb{Z}_p$

### b.a Separable roots

Recall that by Hensel's lemma, if  $r$  and  $p$  are coprime, then an element  $a \in \mathbb{Z}_p$  with zero valuation is a  $r$ th power in  $\mathbb{Z}_p$  if, and only if,  $a_0$  is a  $r$ th power in  $\mathbb{F}_p$ .

Since  $9 = 4^4 = 6^4 = 7^4 = 9^4$  in  $\mathbb{F}_{13}$ ,  $z$  is in fact a 4th power of four elements in  $\mathbb{Z}_{13}$ . As explained in Chapter II, Section 7.1, since 4 and 13 are coprime, we can compute recursively any of these 4th roots. This is done thanks to the `separable_root (a, r)` function which computes a  $r$ th root of  $a$ :

```
Mmx] set_output_order (x, 20);
      t1 == separable_root (z, 4); t2 == separable_root (z, 4);
      t3 == separable_root (z, 4); t4 == separable_root (z, 4);
      mmout << "t1 = " << t1 << ",\n\nt2 = " << t2
      << ",\n\nt3 = " << t3 << ",\n\nt4 = " << t4 << ".";

t1 = 4 + 3 p + 11 p^2 + 9 p^4 + p^5 + 8 p^6 + 7 p^8 + 6 p^9 + 12 p^10 + 3 p^11 + 6 p^12 +
5 p^13 + 7 p^14 + 6 p^16 + 5 p^17 + p^18 + p^19 + O(p^20),

t2 = 9 + 9 p + p^2 + 12 p^3 + 3 p^4 + 11 p^5 + 4 p^6 + 12 p^7 + 5 p^8 + 6 p^9 + 9 p^11 +
6 p^12 + 7 p^13 + 5 p^14 + 12 p^15 + 6 p^16 + 7 p^17 + 11 p^18 + 11 p^19 + O(p^20),

t3 = 6 + 2 p + p^2 + p^3 + 10 p^4 + 12 p^5 + 7 p^6 + 7 p^7 + 5 p^8 + 10 p^9 + 3 p^10 +
10 p^11 + p^12 + 7 p^13 + 2 p^14 + 8 p^15 + 8 p^16 + 7 p^17 + 12 p^18 + 2 p^19 + O(p^20),

t4 = 4 + 3 p + 11 p^2 + 9 p^4 + p^5 + 8 p^6 + 7 p^8 + 6 p^9 + 12 p^10 + 3 p^11 + 6 p^12 +
5 p^13 + 7 p^14 + 6 p^16 + 5 p^17 + p^18 + p^19 + O(p^20).
```

Each call computes randomly one of the roots. Unfortunately, the fourth call returned  $t_4 = t_1$ , while we would like it to return the remaining one, that is  $t_4 = 7 + O(p)$ :

```
Mmx] t4 == separable_root (z, 4);
      mmout << "t4 = " << t4 << ".";

t4 = 7 + 10 p + 11 p^2 + 11 p^3 + 2 p^4 + 5 p^6 + 5 p^7 + 7 p^8 + 2 p^9 + 9 p^10 + 2 p^11 +
11 p^12 + 5 p^13 + 10 p^14 + 4 p^15 + 4 p^16 + 5 p^17 + 10 p^19 + O(p^20).
```

Let us verify now that they all are indeed 4th root of  $z$ . We set the precision up to 10000 and we check that for each  $i$ ,  $1 \leq i \leq 4$ ,  $t_i^4 - z = 0 \pmod{13^{10000}}$ .

```
Mmx] set_output_order (z, 10000);
      mmout << "t1^4-z = " << (t1^4-z) << ",\ntt2^4-z = "
      << (t2^4-z) << ",\n\nt3^4-z = " << (t3^4-z) << ",\ntt4^4-z = "
      << (t4^4-z) << ".";

t1^4-z = O(p^10000),      t2^4-z = O(p^10000),

t3^4-z = O(p^10000),      t4^4-z = O(p^10000).
```

Likewise, since  $4^{25} = 4$  in  $\mathbb{F}_{13}$ ,  $t_1$  is a 25th root in  $\mathbb{Z}_{13}$ . However, there is no other 25th root of 4 in  $\mathbb{F}_{13}$  than 3 itself. So that there is only one element  $u_1 \in \mathbb{Z}_{13}$  such that  $u_1^{25} = t_1$ , its expression up till precision 20 is

```
Mmx] set_output_order (t1, 20); u1 == separable_root (t1, 25);
mmout << "u1 = " << u1 << ".";

u1 = 4 + 6 p + 8 p^2 + 6 p^3 + 9 p^4 + 12 p^5 + 6 p^6 + 7 p^7 + 11 p^8 + 4 p^9 + 7 p^10 + p^11 +
10 p^12 + 5 p^13 + 9 p^14 + 11 p^15 + p^16 + 3 p^17 + 12 p^18 + 10 p^19 + O(p^20).
```

The verification up till precision 10000 gives us

```
Mmx] set_output_order (u1, 10000);
mmout << "u1^25-t1 = " << (u1^25-t1) << ".";

u1^25-t1 = O(p^10000).
```

## b.b $p$ th roots in $\mathbb{Z}_p$

According to Section 7.2, in some favorable cases, if  $a \in \mathbb{Z}_p$ , we can compute a  $p$ th root of  $a$ , which is unique in  $\mathbb{Z}_p$  whenever  $p > 2$ . For instance, let us consider a favorable  $a \in \mathbb{Z}_{65537}$ , that is such that  $a_0^{65537} = a_0 + 65537 a_1 + O(65537^2)$ :

```
Mmx] a : P_adic Modular Integer == p_adic (@p_expansion
(1452^65537+(65537^2)*(12795747464538920304), modulus 65537));
set_output_order (a, 20);
mmout << "a = " << a << ".";

a = 1452 + 52260 p + 45337 p^2 + 49635 p^3 + 45761 p^4 + 3446 p^5 + 23409 p^6 +
7285 p^7 + 2056 p^8 + 20520 p^9 + 58620 p^10 + 196 p^11 + 26660 p^12 + 21951 p^13 +
60415 p^14 + 9757 p^15 + 31425 p^16 + 31575 p^17 + 14102 p^18 + 11450 p^19 + O(p^20).
```

Its 65537th root in  $\mathbb{Z}_{65537}$  is given by a call to the `pth_root` function:

```
Mmx] b == pth_root (a);
mmout << "b = " << b << ", ";

b = 1452 + 25086 p + 63870 p^2 + 53985 p^3 + 62033 p^4 + 14530 p^5 + 19936 p^6 +
35676 p^7 + 22542 p^8 + 46629 p^9 + 8712 p^10 + 23347 p^11 + 19194 p^12 + 9172 p^13 +
18118 p^14 + 9484 p^15 + 11528 p^16 + 55117 p^17 + 7620 p^18 + 54288 p^19 + O(p^20),
```

whose verification is:

```
Mmx] set_output_order (b, 10000);
mmout << "b^65537-a = " << (b^65537-a) << ".";

b^65537-a = O(p^10000).
```

Now, when  $p = 2$ , a favorable  $c \in \mathbb{Z}_2$  is such that  $c = 1 + O(8)$ :

```
Mmx] c : P_adic Modular Integer == p_adic (@p_expansion
(1+8*7638473, modulus 2)); set_output_order (c, 20);
mmout << "c = " << c << ".";

c = 1 + p^3 + p^6 + p^9 + p^10 + p^11 + p^13 + p^14 + p^18 + O(p^20).
```

One of its square root is  $d_1$ :

```
Mmx] d1 == pth_root (c);
mmout << "d1 = " << d1 << ".";

d1 = 1 + p^2 + p^3 + p^4 + p^7 + p^8 + p^9 + p^12 + p^13 + p^14 + p^16 + p^17 + O(p^20).
```

In fact, we made the choice that whenever  $p=2$  and  $c=1+O(8)$ , `pthroot` always returns the root  $d_1$  verifying  $d_1=1+O(4)$ . The other one  $d_2$ , which satisfies  $d_2=3+O(4)$ , is naturally obtained as the opposite of  $d_1$ :

```
Mmx] d2 == -d1;
      mmout << "d2 = " << d2 << ".";

      d2 = 1 + p + p^5 + p^6 + p^10 + p^11 + p^15 + p^18 + p^19 + O(p^20).
```

Let us verify that both  $d_1$  and  $d_2$  are indeed square roots of  $c$ , up till at least precision 1000000:

```
Mmx] set_output_order (d1, 1000000);
      mmout << "d1^2 - c = " << (d1^2-c) << ", \td2^2-c = "
      << (d2^2-c) << ".";
```

⚡ Interrupted

```
Mmx]
```



# Résumé

Contributions à la résolution des systèmes algébriques :  
réduction, localisation, traitement des singularités ; implantations

Cette thèse traite de certains aspects particuliers de la résolution des systèmes algébriques.

Dans un premier temps, nous présentons une façon de minimiser le nombre de variables additives apparaissant dans un système algébrique. Nous utilisons pour cela deux invariants de variété introduits par Hironaka : le faite et la directrice. Dans un second temps, nous proposons une arithmétique rapide, dite détendue, pour les entiers  $p$ -adiques. Cette arithmétique nous permet ensuite de résoudre efficacement un système algébrique à coefficients rationnels localement, c'est-à-dire sur les entiers  $p$ -adiques. En quatrième partie, nous nous intéressons à la factorisation d'un polynôme à deux variables qui est une brique élémentaire pour la décomposition en composantes irréductibles des hypersurfaces. Nous proposons un algorithme réduisant la factorisation du polynôme donné en entrée à celle d'un polynôme dont la taille dense est essentiellement équivalente à la taille convexe-dense de celui donné en entrée. Dans la dernière partie, nous considérons la résolution en moyenne des systèmes algébriques réels. Nous proposons un algorithme probabiliste calculant un zéro approché complexe du système algébrique réel donné en entrée.

**Mots clefs.** Résolution des systèmes algébriques, algorithmes, nombres  $p$ -adiques, factorisation de polynômes.

# Abstract

Contributions to algebraic system solving:  
reduction, localization, singularities handling; implementations

This PhD thesis deals with some particular aspects of the algebraic systems resolution.

Firstly, we introduce a way of minimizing the number of additive variables appearing in an algebraic system. For this, we make use of two invariants of variety introduced by Hironaka: the ridge and the directrix. Then, we propose fast arithmetic routines, the so-called relaxed routines, for  $p$ -adic integers. These routines allow us, then, to solve efficiently an algebraic system with rational coefficients locally, *i.e.* over the  $p$ -adic integers. In a fourth part, we are interested in the factorization of a bivariate polynomial, which is at the root of the decomposition of hypersurfaces into irreducible components. We propose an algorithm reducing the factorization of the input polynomial to that of a polynomial whose dense size is essentially equivalent to the convex-dense size of the input polynomial. In the last part, we consider real algebraic systems solving in average. We design a probabilistic algorithm computing an approximate complex zero of the real algebraic system given as input.

**Key words.** Algebraic systems solving, algorithms,  $p$ -adic numbers, polynomial factorization