Fast amortized multi-point evaluation[*][†]

JORIS VAN DER HOEVEN[abc], GRÉGOIRE LECERF[bd]

[a]. CNRS (UMI 3069, PIMS)
Department of Mathematics
Simon Fraser University
8888 University Drive
Burnaby, British Columbia
V5A 1S6, Canada

[b]. CNRS, École polytechnique, Institut Polytechnique de Paris
Laboratoire d'informatique de l'École polytechnique (LIX, UMR 7161)
1, rue Honoré d'Estienne d'Orves
Bâtiment Alan Turing, CS35003
91120 Palaiseau, France

[c]. *Email:* `vdhoeven@lix.polytechnique.fr`
[d]. *Email:* `lecerf@lix.polytechnique.fr`

Preliminary version of March 14, 2020

The efficient evaluation of multivariate polynomials at many points is an important operation for polynomial system solving. Kedlaya and Umans have recently devised a theoretically efficient algorithm for this task when the coefficients are integers or when they lie in a finite field. In this paper, we assume that the set of points where we need to evaluate is fixed and "sufficiently generic". Under these restrictions, we present a quasi-optimal algorithm for multi-point evaluation over general fields. We also present a quasi-optimal algorithm for the opposite interpolation task.

1. INTRODUCTION

Let $\mathbb{K}$ be an effective field, so that we have algorithms for the field operations. Given a polynomial $P \in \mathbb{K}[x_1, \ldots, x_n]$ and a tuple $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_d) \in (\mathbb{K}^n)^d$ of points, the computation of $P(\boldsymbol{\alpha}) = (P(\alpha_1), \ldots, P(\alpha_d)) \in \mathbb{K}^d$ is called the problem of *multi-point evaluation*. The converse problem is called *interpolation* and takes a candidate support of P as input.

This problem naturally relates to several areas of applied algebra, including polynomial system solving, since we may use it to verify whether all points in a given set are solutions to a system of polynomial equations. In [16], it has even be shown that efficient algorithms for multi-point evaluation lead to efficient algorithms for polynomial system solving. As an other more specific application, bivariate polynomial evaluation intervenes in computing generator matrices of geometric error correcting codes [20].

An important particular case of interpolation concerns polynomials with prescribed supports and that vanish at a given set of points. This happens in list decoding algorithms for Reed–Solomon error correcting codes; see recent complexity bounds in [7]. In the bivariate case, this task also occurs in the Brill–Noether strategy for computing Riemann–Roch spaces; see recent advances in [2]. Further applications can be found in [24].

*. This paper is part of a project that has received funding from the French "Agence de l'innovation de défense".

†. This article has been written using GNU $\mathrm{T_EX_{MACS}}$ [18].

1.1. Related work

In the univariate case when $n=1$, it is well known that one may use so-called "remainder trees" to compute multi-point evaluations in quasi-optimal time [3, 9, 23]. More precisely, if $\mathsf{M}(d)$ stands for the cost to multiply two univariate polynomials of degree $<d$ (in terms of the number of field operations in $\mathbb{K}$), then the multi-point evaluation of a polynomial of degree $<d$ at d points can be computed in time $O(\mathsf{M}(d)\log d)$. A similar complexity bound holds for the opposite operation of *interpolation*. The constants hidden in the latter "O" have been studied in [5]. Recently, and under suitable technical assumptions, it has been shown that the cost of univariate multi-point evaluation (and interpolation) drops to $O(\mathsf{M}(d)\log d/\log\log d)$ if the set of evaluation points is fixed [13]. In this case, we do not count the cost of certain precomputations that only depend on the evaluation points and not on the polynomials to evaluate. We may also speak about the "amortized cost" of multi-point evaluation.

In the multivariate case when $n\geqslant 2$, no quasi-optimal algorithms are currently known for multi-point evaluation. From a theoretical point of view, the best bit-complexity bounds are due to Kedlaya and Umans, in the special cases when the coefficients of our polynomials are modular integers or when they lie in a finite field [19]. They also related the problem to other important operations such as modular composition and power projection. Their results have recently been refined in [17]. Over general fields and for $n=2$, the best known bound is $O(\deg_{x_1} P\,(\deg_{x_2} P)^{1.667})$; due to Nüsken and Ziegler [26].

The multivariate interpolation problem turns out to be more intricate than evaluation, and fewer efficient algorithms are known. Using naive linear algebra, one may design polynomial time algorithms, but with costs that are higher than quadratic in d. To our knowledge no interpolation method has been designed in the vein of the Kedlaya–Umans evaluation algorithms. Several methods are surveyed in [10] for instance, but, here, we will focus on symbolic approaches and their asymptotical complexity bounds.

Concerning the computation of polynomials that vanish at a given set of points, with suitable constraints on the supports, early methods can be found in [1, 22, 24]: Gröbner bases of the ideal made of these polynomials are obtained with cost at least cubic in d. After a generic change of coordinates the lexicographic basis satisfies the "shape lemma" and it can be computed in softly linear time by means of univariate interpolations. In other words, our set of points is the solution set of a system $\chi(x_1)=0$ and $x_i=v_i(x_1)$ for $i=2,\ldots,n$, where $\deg\chi=d$ and $\deg v_i<d$. From χ and the v_i, a Gröbner basis for any other monomial ordering can be recovered with $\tilde{O}(d^{\omega})$ field operations thanks to the algorithm designed in [8].

In the bivariate case, with generic coordinates, from the latter univariate parametrization, and given $\delta_1\leqslant d$, we may compute a basis of the $\mathbb{K}[x_1]$-module of polynomials of degree $<\delta_1$ in x_2 that vanish at $\alpha_1,\ldots,\alpha_d$. Indeed this reduces to computing a basis of the kernel of the map

$$\begin{array}{rcl} \mathbb{K}[x_1][x_2]_{<\delta_1} & \to & \mathbb{K}[x_1]/(\chi(x_1)) \\ a_0(x_1)+\cdots+a_{\delta_1-1}(x_1)\,x_2^{\delta_1-1} & \mapsto & a_0(x_1)+\cdots+a_{\delta_1-1}(x_1)\,v_2(x_1)^{\delta_1-1}. \end{array}$$

This kernel is a free $\mathbb{K}[X_1]$-module of rank δ_1. The basis in Popov form can be computed with $\tilde{O}(\delta_1^{\omega-1}d)$ operations in $\mathbb{K}$; this complexity bound is due to Neiger [25]. Taking $\delta_1\asymp\sqrt{d}$, the latter cost rewrites $\tilde{O}(d^{(\omega+1)/2})$. The bivariate interpolation problem can be solved with the same kind of complexity bound by means of linear algebra algorithms that exploit displacement ranks [4].

In a more general setting, one may consider evaluation at "multisets of points" in the sense that values of polynomials and of certain of its derivatives are involved. The converse problem is usually called *Hermite interpolation*, so values for the polynomial and its derivative are prescribed. We will not address these extended problems in the present paper.

1.2. Our contributions

In this paper, we turn our attention to the amortized cost of multivariate multi-point evaluation. Given a "sufficiently generic" tuple $\boldsymbol{\alpha}$ of evaluation points, we first construct a suitable "evaluation tree" that is similar to the remainder trees from the univariate case. After this precomputation, we next show how to evaluate polynomials at $\boldsymbol{\alpha}$ in quasi-optimal time. For instance, for a fixed dimension n, a sufficiently large base field $\mathbb{K}$, and a polynomial P of partial degrees $\deg_{x_i} P < \sqrt[n]{d}$ for $i = 1, \ldots, n$, we show how to compute $P(\boldsymbol{\alpha})$ in time $O(\mathsf{M}(d) \log^3 d)$. We also show how to do the opposite operation of interpolation with a similar complexity. It can be shown that the "constant factors" in the big-Oh hide a factorial dependence in n.

Our algorithms rely on suitable genericity hypotheses that ensure the existence of Gröbner bases of a specific shape for the vanishing ideal $I_{\boldsymbol{\alpha}}$ at the evaluation points. This will be detailed in section 3. Another key ingredient is an algorithm from [14] for the relaxed reduction of polynomials with respect to such Gröbner bases or more general sets of polynomials. In this paper, all reductions will be done with respect to so-called "axial bases", which provide sufficiently good approximations of the actual Gröbner bases of $I_{\boldsymbol{\alpha}}$. This will be detailed in section 4.

Having an efficient algorithm for reducing polynomials with respect to $I_{\boldsymbol{\alpha}}$, we may use it as a generalization of Euclidean division. In sections 5 and 6, this allows us to generalize the remainder tree technique to higher dimensions, and establish our complexity bounds for amortized multi-point evaluation and interpolation. It is also noteworthy that the necessary precomputations can be done using linear algebra in time $O(d^\omega)$, where $\omega > 2$ is a feasible exponent for matrix multiplication.

When comparing our new amortized bound $O(\mathsf{M}(d) \log^3 d)$ with the traditional bound $O(\mathsf{M}(d) \log d)$ in the univariate case, we note that we lose a factor $\log^2 d$. This is due to the fact that we rely on relaxed computations for our polynomial reductions. With a bit of work, a factor $\log d$ might be removed by exploiting the fact that our polynomials are not really sparse, so that we may take $\mathsf{SM}(s) = O(\mathsf{M}(s))$ in the proof of Theorem 6 (under our assumption that n is fixed and with the notation SM defined in [14]). It is plausible that the second logarithmic factor can be further reduced using techniques from [12] or a clever Newton iteration.

Our genericity assumptions can also be weakened significantly: with the notations from section 4, it is essentially sufficient to assume that $|\mathfrak{R}_d| = O(d)$. Another interesting question is whether the cost $O(d^\omega)$ of the precomputations can be lowered. Under additional genericity assumptions, this is indeed possible when $n = 2$: using the probabilistic algorithm of type Las Vegas from [4], the precomputations take an expected number of $O(d^{(\omega+1)/2 + o(1)})$ field operations.

2. PRELIMINARIES

For complexity analyses, we will only consider algebraic complexity models (such as computation trees). In other words we will simply count numbers of arithmetic operations and zero-tests in $\mathbb{K}$.

Throughout this paper, we assume ω to be a feasible exponent for matrix multiplication with $\omega > 2$. This means that two $n \times n$ matrices in $\mathbb{K}^{n \times n}$ can be multiplied in time $O(n^{\omega})$. Le Gall has shown in [21] that one may take $\omega < 2.373$.

We denote by $\mathsf{M}(d)$ the time that is needed to compute a product PQ of two polynomials $P, Q \in \mathbb{K}[x]$ of degree $<d$. We make the usual assumption that $\mathsf{M}(d)/d$ is non-decreasing as a function of d. It is also convenient to assume that $\mathsf{M}(kd) = O(k\mathsf{M}(d))$ for $k = O(\log d)$. Using a variant of Schönhage–Strassen's algorithm [6, 28, 29], it is well known that $\mathsf{M}(d) = O(d \log d \log \log d)$. If we restrict our attention to fields $\mathbb{K}$ of positive characteristic, then we may take $\mathsf{M}(d) = O(d \log d \, 4^{\log^* n})$ [11].

In order to use the extended multivariate reduction algorithms from [14], sparse polynomial arithmetic is needed. A *sparse representation* of a polynomial F in $\mathbb{K}[x_1, \ldots, x_n]$ is a data structure that stores the set of the non-zero terms of F. Each such term is a pair made of a coefficient and a degree vector. In an algebraic complexity model the bit size of the exponents counts for free, so the relevant size of such a polynomial is the cardinality of its support.

Consider two polynomials F and G of $\mathbb{K}[x_1, \ldots, x_n]$ in sparse representation. An extensive literature exists on the general problem of multiplying F and G; see [27] for a recent survey. For the purposes of this paper, a superset $\mathcal{S}$ for the support of FG will always be known. Then we define $\mathsf{SM}(s)$ to be the cost to compute FG, where s is the maximum of the sizes of such a superset $\mathcal{S}$ and the supports of F and G. Under suitable assumptions, the following proposition will allow us to take $\mathsf{SM}(s) = O(\mathsf{M}(s) \log s)$ in our multivariate evaluation and interpolation algorithms.

PROPOSITION 1. *Let $\pi_1, \ldots, \pi_n$ be positive integers and let θ in $\mathbb{K}$ be of multiplicative order at least $\pi := \pi_1 \cdots \pi_n$.*

i. The set $\mathcal{P}$ made of the products $\theta^{e_1} \theta^{e_2 \pi_1} \cdots \theta^{e_n \pi_1 \cdots \pi_{n-1}}$ for $(e_1, \ldots, e_n) \in \prod_{i=1}^{n} \{0, \ldots, \pi_i - 1\}$ can be computed using $O(\pi)$ operations in $\mathbb{K}$.

ii. Let F and G be in $\mathbb{K}[x_1, \ldots, x_n]$, in sparse representation, and let $\mathcal{S}$ be a superset of the support of FG. Assume that $\deg_{x_i}(FG) < \pi_i$ for $i = 1, \ldots, n$, and that $\mathcal{P}$ has been precomputed. Then the product FG can be computed using $O(\mathsf{M}(s) \log s)$ operations in $\mathbb{K}$, where s denotes the maximum of the sizes of $\mathcal{S}$ and the supports of F and G.

Proof. The first statement is straightforward. The second one is simply adapted from [15, Proposition 6]. □

Computing an element of sufficiently large multiplicative order depends on the ground field $\mathbb{K}$. In characteristic zero, any integer different from 0 and 1 will do. For finite fields $\mathbb{F}_q$ of characteristic $p > 0$, working in an algebraic extension $\mathbb{F}_{q^\lambda}$ yields elements of order up to $q^\lambda - 1$. In the context of Proposition 1, we may take $\lambda = O(\log \pi / \log q)$. In infinite fields $\mathbb{K}$ of positive charactersitic $p > 0$, elements of arbitrarily large orders exist, but they may be hard to access without prior knowledge. However, this is mostly a theoretical problem: in practice, we usually either know a transcendental element of infinite order or a way to construct arbitrarily large finite fields inside $\mathbb{K}$.

3. GRÖBNER BASES FOR GENERIC SETS OF POINTS

For variables $\boldsymbol{x} = (x_1, \ldots, x_n)$ and exponents $\boldsymbol{i} = (i_1, \ldots, i_n) \in \mathbb{N}^n$, we define $\boldsymbol{x}^{\boldsymbol{i}} := x_1^{i_1} \cdots x_n^{i_n}$, $\mathfrak{M} := \{\boldsymbol{x}^{\boldsymbol{i}} : \boldsymbol{i} \in \mathbb{N}^n\}$, and $\mathbb{K}[\boldsymbol{x}] := \mathbb{K}[x_1, \ldots, x_n]$. The monoid $\mathfrak{M}$ comes with a natural partial ordering $\leqslant$ that is defined by

$$\boldsymbol{x}^{\boldsymbol{i}} \leqslant \boldsymbol{x}^{\boldsymbol{j}} \Leftrightarrow i_1 \leqslant j_1 \wedge \cdots \wedge i_n \leqslant j_n.$$

We also assume that we fixed a total admissible ordering $\preccurlyeq$ on $\mathfrak{M}$ that extends $\leqslant$ and which turns $\mathfrak{M}$ into a totally ordered monoid. Given a polynomial

$$P = \sum_{\mathfrak{m} \in \mathfrak{M}} P_{\mathfrak{m}} \mathfrak{m} \in \mathbb{K}[x],$$

we call $\operatorname{supp} P := \{\mathfrak{m} \in \mathfrak{M} : P_{\mathfrak{m}} \neq 0\}$ its *support*. If $P \neq 0$, then we define

$$\mathfrak{d}_P := \max_{\preccurlyeq} \operatorname{supp} f$$

to be the *dominant monomial* of P.

3.1. Polynomials that vanish on a finite set of points

Let $\mathfrak{F} = \{\mathfrak{m}_1, \ldots, \mathfrak{m}_d\}$ be a finite set of monomials. Given a polynomial $P = c_1 \mathfrak{m}_1 + \cdots + c_d \mathfrak{m}_d$ and a finite tuple of points $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_d) \in (\mathbb{K}^n)^d$, we have

$$\begin{pmatrix} P(\alpha_1) \\ \vdots \\ P(\alpha_d) \end{pmatrix} = \begin{pmatrix} \mathfrak{m}_1(\alpha_1) & \cdots & \mathfrak{m}_d(\alpha_1) \\ \vdots & & \vdots \\ \mathfrak{m}_1(\alpha_d) & \cdots & \mathfrak{m}_d(\alpha_d) \end{pmatrix} \begin{pmatrix} c_1 \\ \vdots \\ c_d \end{pmatrix}.$$

The set of tuples of points for which the matrix

$$M_{\boldsymbol{\alpha},\mathfrak{F}} := \begin{pmatrix} \mathfrak{m}_1(\alpha_1) & \cdots & \mathfrak{m}_d(\alpha_1) \\ \vdots & & \vdots \\ \mathfrak{m}_1(\alpha_d) & \cdots & \mathfrak{m}_d(\alpha_d) \end{pmatrix}$$

is non-invertible forms a proper Zariski closed subset of $(\mathbb{K}^n)^d$. If $\mathbb{K}$ is algebraically closed, then its open complement is actually non-empty:

LEMMA 2. *Assume that $\mathbb{K}$ is algebraically closed. Then there exists a tuple of points $\boldsymbol{\alpha} \in (\mathbb{K}^n)^d$ for which $M_{\boldsymbol{\alpha},\mathfrak{F}}$ is invertible.*

Proof. Let $t = (t_1, \ldots, t_n) \in \mathbb{K}^n$ be a point such that $\mathfrak{m}_1(t), \ldots, \mathfrak{m}_d(t)$ are pairwise distinct; such a point t exists since $\mathbb{K}$ is algebraically closed. Now take $\alpha_k = (t_1^k, \ldots, t_n^k)$ for $k = 1, \ldots, d$. Then have

$$M_{\boldsymbol{\alpha},\mathfrak{F}} = \begin{pmatrix} \mathfrak{m}_1(\alpha_1) & \cdots & \mathfrak{m}_d(\alpha_1) \\ \vdots & & \vdots \\ \mathfrak{m}_1(\alpha_1)^d & \cdots & \mathfrak{m}_d(\alpha_1)^d \end{pmatrix},$$

whence $M_{\boldsymbol{\alpha},\mathfrak{F}}$ is an invertible Vandermonde matrix. □

The lemma shows that $M_{\boldsymbol{\alpha},\mathfrak{F}}$ is invertible for a "generic" tuple of points $\boldsymbol{\alpha} \in (\mathbb{K}^n)^d$. Assume from now on that this is indeed the case and let us write $I_{\boldsymbol{\alpha}}$ for the ideal of polynomials $P \in \mathbb{K}[x]$ such that $P(\alpha_1) = \cdots = P(\alpha_d) = 0$. For any other monomial $\mathfrak{n} \in \mathfrak{M} \setminus \mathfrak{F}$ and $P = \mathfrak{n} - (c_1 \mathfrak{m}_1 + \cdots + c_d \mathfrak{m}_d)$, we have

$$\begin{pmatrix} P(\alpha_1) \\ \vdots \\ P(\alpha_d) \end{pmatrix} = \begin{pmatrix} \mathfrak{n}(\alpha_1) \\ \vdots \\ \mathfrak{n}(\alpha_d) \end{pmatrix} - M_{\boldsymbol{\alpha},\mathfrak{F}} \begin{pmatrix} c_1 \\ \vdots \\ c_d \end{pmatrix},$$

whence $P \in I_{\boldsymbol{\alpha}}$ if and only if

$$\begin{pmatrix} c_1 \\ \vdots \\ c_d \end{pmatrix} = M_{\boldsymbol{\alpha},\mathfrak{F}}^{-1} \begin{pmatrix} \mathfrak{n}(\alpha_1) \\ \vdots \\ \mathfrak{n}(\alpha_d) \end{pmatrix}. \tag{1}$$

This shows that $\mathfrak{m}_1 + I_{\boldsymbol{\alpha}}, \ldots, \mathfrak{m}_d + I_{\boldsymbol{\alpha}}$ form a basis for the quotient algebra $\mathbb{K}[x] / I_{\boldsymbol{\alpha}}$ and it provides us with a formula to express $\mathfrak{n} + I_{\boldsymbol{\alpha}}$ in terms of these basis elements.

3.2. Gröbner bases

Let $1=\mathfrak{m}_1 \prec \mathfrak{m}_2 \prec \mathfrak{m}_3 \prec \cdots$ be such that $\mathfrak{M}_d := \{\mathfrak{m}_1, \ldots, \mathfrak{m}_d\}$ is the set of d smallest elements of $\mathfrak{M}$ for each $d \in \mathbb{N}$ and with respect to the total ordering $\preccurlyeq$. We define $\mathfrak{G}_d$ to be the set of smallest elements of the complement $\mathfrak{M} \setminus \mathfrak{M}_d$ for $\leqslant$. By Dickson's lemma, this set is finite and it forms an antichain for $\leqslant$. Let $\boldsymbol{\alpha} \in (\mathbb{K}^n)^d$ be a generic tuple of points in the sense that $M_{\boldsymbol{\alpha},\mathfrak{M}_d}$ is invertible.

PROPOSITION 3. *For each* $\mathfrak{n} \in \mathfrak{G}_d$*, let*

$$G_{\boldsymbol{\alpha},\mathfrak{n}} \;=\; \mathfrak{n} - (c_1 \mathfrak{m}_1 + \cdots + c_d \mathfrak{m}_d), \tag{2}$$

where $c_1, \ldots, c_d$ *satisfy* (1) *for* $\mathfrak{F} = \mathfrak{M}_d$*. Then* $\boldsymbol{G}_{\boldsymbol{\alpha}} := \{G_{\boldsymbol{\alpha},\mathfrak{n}} : \mathfrak{n} \in \mathfrak{G}_d\}$ *forms the reduced Gröbner basis of* $I_{\boldsymbol{\alpha}}$ *with respect to* $\preccurlyeq$*.*

Proof. Let $\mathfrak{D} \subseteq \mathfrak{M}$ be the set of dominant monomials of non-zero elements of $I_{\boldsymbol{\alpha}}$. Given $i \in \{1, \ldots, d\}$, we have $\mathfrak{m}_i \notin \mathfrak{D}$: otherwise, $P = \mathfrak{m}_i + c_{i-1} \mathfrak{m}_{i-1} + \cdots + c_1 \mathfrak{m}_1 \in I_{\mathrm{A}}$ for certain $c_1, \ldots, c_{i-1} \in \mathbb{K}$, which is impossible since $M_{\boldsymbol{\alpha},\mathfrak{M}_d}$ is invertible. Conversely, (1) implies that $\mathfrak{n} \in \mathfrak{D}$ for all $\mathfrak{n} \in \mathfrak{M} \setminus \mathfrak{M}_d$, whence $\mathfrak{D} = \mathfrak{M} \setminus \mathfrak{M}_d$.

Now it is well known that $\mathfrak{D}$ is a finite segment of $\mathfrak{M}$ for $\leqslant$ and that each minimal element $\mathfrak{n}$ corresponds to exactly one polynomial $R_{\mathfrak{n}}$ in the reduced Gröbner basis with dominant monomial $\mathfrak{d}_{R_{\mathfrak{n}}} = \mathfrak{n}$. Now such a reduced polynomial is by definition of the form $R_{\mathfrak{n}} = \mathfrak{n} + \Delta$ with $\operatorname{supp} \Delta \subseteq \mathfrak{M} \setminus \mathfrak{D} = \mathfrak{M}_d$. But (2) shows how to compute the unique polynomial $R_{\mathfrak{n}} = G_{\boldsymbol{\alpha},\mathfrak{n}}$ of that form. □

Example 4. Let $\preccurlyeq$ be a graded ordering for $n \geqslant 2$. For any $d \geqslant n+1$, we have $\mathfrak{m}_1 = 1$ and $\mathfrak{m}_{i+1} = x_i$ for $i = 1, \ldots, n$. The $(i+1)$-th column of $M_{\boldsymbol{\alpha},\mathfrak{F}}$ contains the i-th coordinates of the points of $\boldsymbol{\alpha}$. If these columns are not linearly independent then $M_{\boldsymbol{\alpha},\mathfrak{F}}$ is not invertible. If A is any $n \times n$ invertible matrix over $\mathbb{K}$, and if $A(\boldsymbol{\alpha})$ denotes $A\,\alpha_1, \ldots, A\,\alpha_d$ then the columns from 2 to $n+1$ of $M_{A(\boldsymbol{\alpha}),\mathfrak{F}}$ are linearly dependent, so $M_{A(\boldsymbol{\alpha}),\mathfrak{F}}$ is not invertible. This example illustrates that the genericity of a tuple of points $\boldsymbol{\alpha} \in (\mathbb{K}^n)^d$ cannot be necessarily recovered after a linear change of the coordinates.

4. RELAXED REDUCTION WITH RESPECT TO AXIAL BASES

Let $\mathfrak{M} = \{\boldsymbol{x}^{\boldsymbol{i}} : \boldsymbol{i} \in \mathbb{N}^n\}$ be as in the previous section, with a total admissible ordering $\preccurlyeq$, and let $\boldsymbol{\alpha} \in (\mathbb{R}^n)^d$ be a generic tuple of points. We need a way to efficiently reduce polynomials $P \in \mathbb{K}[\boldsymbol{x}]$ with respect to the Gröbner basis $\boldsymbol{G}_{\boldsymbol{\alpha}}$. Since the entire Gröbner basis can be voluminous to store, we will only reduce with respect to a special subset $\boldsymbol{A}_{\boldsymbol{\alpha}}$ of "axial" basis elements. This also requires us to work with respect to a weighted total degree ordering $\preccurlyeq$.

4.1. Axial bases

For $i = 1, \ldots, d$, we define

$$\delta_{d,i} \;:=\; \min \{k \in \mathbb{N} : x_i^k \notin \mathfrak{M}_d\},$$

so that $\boldsymbol{G}_{\boldsymbol{\alpha}}$ contains a unique element $A_{\boldsymbol{\alpha},i} := G_{\boldsymbol{\alpha},\mathfrak{a}_i}$ with dominant monomial $\mathfrak{a}_i := x_i^{\delta_{d,i}}$. We define $\boldsymbol{A}_{\boldsymbol{\alpha}} := \{A_{\boldsymbol{\alpha},i} : i = 1, \ldots, n\}$ to be the *axial basis* of $\boldsymbol{A}_{\boldsymbol{\alpha}}$. Although this set is not a "basis" of $I_{\boldsymbol{\alpha}}$, it forms a sufficiently good approximation of $\boldsymbol{G}_{\boldsymbol{\alpha}}$ for the purposes of this paper. We define

$$\mathfrak{R}_d \;:=\; \{\boldsymbol{x}^{\boldsymbol{i}} : i_1 < \delta_{d,1} \wedge \cdots \wedge i_n < \delta_{d,n}\}$$

to be the set of monomials that are not divisible by any of the monomials $\mathfrak{a}_1, \ldots, \mathfrak{a}_n$. We also define Red_d to be the $\mathbb{K}$-vector spaced spanned by the elements of $\mathfrak{R}_d$.

4.2. Weighted total degree orderings

In all what follows, we assume that the ordering is graded by a weighted total degree. This means that there exist positive real weights $w_1, \ldots, w_n > 0$ such that for all $\boldsymbol{x}^{\boldsymbol{i}}, \boldsymbol{x}^{\boldsymbol{j}} \in \mathfrak{M}$, we have

$$\boldsymbol{w} \cdot \boldsymbol{i} < \boldsymbol{w} \cdot \boldsymbol{j} \implies \boldsymbol{x}^{\boldsymbol{i}} \prec \boldsymbol{x}^{\boldsymbol{j}}.$$

Here $\boldsymbol{w} = (w_1, \ldots, w_n)$ and "$\cdot$" stands for the dot product: $\boldsymbol{w} \cdot \boldsymbol{i} = w_1 i_1 + \cdots + w_n i_n$. Let

$$s := \max \{\boldsymbol{w} \cdot \boldsymbol{i} : \boldsymbol{x}^{\boldsymbol{i}} \in \mathfrak{M}_d\}.$$

Then

$$\{\boldsymbol{x}^{\boldsymbol{i}} : \boldsymbol{w} \cdot \boldsymbol{i} < s\} \subsetneq \mathfrak{M}_d \subseteq \{\boldsymbol{x}^{\boldsymbol{i}} : \boldsymbol{w} \cdot \boldsymbol{i} \leqslant s\}.$$

Geometrically speaking, the exponents $\boldsymbol{i}$ with $\boldsymbol{x}^{\boldsymbol{i}} \in \mathfrak{M}_d$ correspond to lattice points inside or on the boundary of the simplex with vertices $(0,\ldots,0)$, $(s/w_1, 0, \ldots, 0)$, $(0, s/w_2, 0, \ldots, 0)$, ..., $(0, \ldots, 0, s/w_n)$. For fixed n and large d (whence s), it follows that

$$\begin{aligned} d &\sim \frac{s^n}{n!\, w_1 \cdots w_n} \\ \delta_{d,i} &\sim \frac{s}{w_i} \sim \frac{\sqrt[n]{n!\, w_1 \cdots w_n}}{w_i} \qquad (i = 1, \ldots, n) \\ |\mathfrak{R}_d| &\sim \frac{s^n}{w_1 \cdots w_n} \sim n!\, d, \end{aligned}$$

where $|\mathfrak{R}_d|$ stands for the cardinality of $\mathfrak{R}_d$. In the remainder of this paper, we assume that n and $w_1, \ldots, w_n$ have been fixed once and for all. Our asymptotic complexity estimates hold for large d and under this assumption.

Remark 5. If $w_1 = \cdots = w_n = 1$, then one may take $\preccurlyeq$ to be the usual graded reverse lexicographic ordering. In that case, the $\delta_{d,i}$ are of the same order of magnitude and $\mathfrak{R}_d$ is approximately a hypercube. Considering general weights gives us more flexibility in the choice of $\preccurlyeq$ and allows us to deal with more general hyperrectangular supports.

4.3. Relaxed reduction

Given a polynomial $P \in \mathbb{K}[\boldsymbol{x}]$, an *extended reduction* of P with respect to $A_{\boldsymbol{\alpha},1}, \ldots, A_{\boldsymbol{\alpha},n}$ is a relation

$$P = Q_1 A_{\boldsymbol{\alpha},1} + \cdots + Q_n A_{\boldsymbol{\alpha},n} + R, \tag{3}$$

with $Q_1, \ldots, Q_n \in \mathbb{K}[\boldsymbol{x}]$ and $R \in \mathrm{Red}_d$. Extended reductions are computed by reducing P with respect to $A_{\boldsymbol{\alpha},i}$ as long as there exists an index i for which $\mathfrak{d}_{A_{\boldsymbol{\alpha},i}}$ divides $\mathfrak{d}_P$. There are many ways to compute extended reductions of P depending on the way we chose the index i in case of multiple options. If we always privilege the lowest possible index i, then the process is deterministic and we call (3) "the" extended reduction of P with respect to $\boldsymbol{A}_{\boldsymbol{\alpha}}$. In that case, we define $P \operatorname{rem} \boldsymbol{A}_{\boldsymbol{\alpha}} := R$ and call it the *remainder* of P with respect to $\boldsymbol{A}_{\boldsymbol{\alpha}}$. Using relaxed evaluation, this remainder can be computed efficiently:

THEOREM 6. *Let $\pi_1 \geqslant \delta_{d,1}, \ldots, \pi_n \geqslant \delta_{d,n}$ be integers, let $\pi := \pi_1 \cdots \pi_n$, and let $P \in \mathbb{K}[\boldsymbol{x}]$ be such that $\deg_{x_i} P < \pi_i$ for $i = 1, \ldots, n$. Then an extended reduction (3) can be computed in time*

$$O(\mathsf{M}(\pi) \log^2 \pi),$$

whenever an element of multiplicative order $\geqslant (n+1)!\, \pi$ is given in $\mathbb{K}$.

Proof. Without loss of generality, we may assume that

$$w_1 \pi_1 \leqslant \cdots \leqslant w_n \pi_n.$$

We use the reduction algorithm from [14] with the reduction strategy that always reduces with respect to the axial element $A_{\boldsymbol{\alpha},k}$ for which k is minimal. Then we claim that the supports of the successive reductions of P are all contained in the set

$$\mathfrak{S} := \{\boldsymbol{x}^{\boldsymbol{i}} : \forall j \in \{1,\ldots,n\}, w_1 i_1 + \cdots + w_j i_j < w_1 \pi_1 + \cdots + w_j \pi_j + w_j \delta_{d,j}\}.$$

Indeed, let $\boldsymbol{x}^{\boldsymbol{e}} \in \mathfrak{S}$, let k be minimal such that $x_k^{\delta_{d,k}}$ divides $\boldsymbol{x}^{\boldsymbol{e}}$, and consider a monimial $\boldsymbol{x}^{\boldsymbol{c}}$ in the support of $T := x_k^{-\delta_{d,k}} \boldsymbol{x}^{\boldsymbol{e}} A_{\boldsymbol{\alpha},k}$. It suffices to show that $\boldsymbol{x}^{\boldsymbol{c}} \in \mathfrak{S}$. Let $\boldsymbol{e}'$ be such that $\boldsymbol{x}^{\boldsymbol{e}'} = x_k^{-\delta_{d,k}} \boldsymbol{x}^{\boldsymbol{e}}$. For $j = 1, \ldots, k-1$, the relations $\boldsymbol{x}^{\boldsymbol{e}'} | \boldsymbol{x}^{\boldsymbol{c}}$ and $\boldsymbol{x}^{\boldsymbol{c}-\boldsymbol{e}'} \prec x_j^{\delta_{d,j}}$ imply

$$\begin{aligned} w_1 (c_1 - e_1) + \cdots + w_j (c_j - e_j) &= w_1 (c_1 - e_1') + \cdots + w_j (c_j - e_j') \\ &\leqslant w_1 (c_1 - e_1') + \cdots + w_n (c_n - e_n') < w_j \delta_{d,j}. \end{aligned}$$

Now $e_1 < \delta_{d,1} \leqslant \pi_1, \ldots, e_{k-1} < \delta_{d,k-1} \leqslant \pi_{k-1}$, whence

$$w_1 c_1 + \cdots + w_j c_j < w_1 e_1 + \cdots + w_j e_j + w_j \delta_{d,j} < w_1 \pi_1 + \cdots + w_j \pi_j + w_j \delta_{d,j}.$$

For $j = k, \ldots, n$, we directly have

$$w_1 c_1 + \cdots + w_j c_j \leqslant w_1 e_1 + \cdots + w_j e_j < w_1 \pi_1 + \cdots + w_j \pi_j + w_j \delta_{d,j}.$$

Having shown our claim, we next observe that $w_k e_k < k w_k \pi_k + w_k \delta_{d,k}$ for all $\boldsymbol{x}^{\boldsymbol{e}} \in \mathfrak{S}$ and $k = 1, \ldots, n$, whence $e_k < (k+1) \pi_k$ and $e_1 \cdots e_n < (n+1)! \pi$. In particular, the size of the support of the $Q_i A_{\boldsymbol{\alpha},i}$ and R in the extended reduction (3) is bounded by $(n+1)! \pi$. The theorem now becomes a consequence of [14, Theorem 4] and Proposition 1, by taking $\mathsf{SM}(s) := O(\mathsf{M}(s) \log s)$. □

Remark 7. If $\mathbb{K}$ is the finite field $\mathbb{F}_q$ and if $q - 1 < (n+1)! \pi$, then we can apply Theorem 6 over an algebraic extension $\mathbb{F}_{q^\lambda}$ of degree

$$\lambda := \lceil \log ((n+1)! \pi) / \log q \rceil.$$

Constructing this extension can be done using $\tilde{O}(\sqrt{\pi})$ operations in $\mathbb{K}$ by [30, Theorem 3.2]. The primitive root of $\mathbb{F}_{q^\lambda}^{\times}$ can be obtained in negligible expected time using a probabilistic algorithm of Las Vegas type. Then the complexity bound in Theorem 6 becomes

$$O\left(\mathsf{M}(\pi) \frac{\log^3 \pi}{\log q}\right).$$

5. MULTI-POINT EVALUATION

The fast algorithms for multi-point evaluation of univariate polynomials make use of the technique of *remainder trees* [3, 9, 23]. For instance, the remainder tree for four points $\alpha_1, \alpha_2, \alpha_3, \alpha_4 \in \mathbb{K}$ is the labeled binary tree given by

$$\begin{tikzcd} & & & (x-\alpha_1)(x-\alpha_2)(x-\alpha_3)(x-\alpha_4) \arrow[dll, dash] \arrow[drr, dash] & & & \\ & (x-\alpha_1)(x-\alpha_2) \arrow[dl, dash] \arrow[dr, dash] & & & & (x-\alpha_3)(x-\alpha_4) \arrow[dl, dash] \arrow[dr, dash] & \\ x-\alpha_1 & & x-\alpha_2 & & x-\alpha_3 & & x-\alpha_4 \end{tikzcd} \tag{4}$$

Given a polynomial $P \in \mathbb{K}[x]$ to evaluate at these four points, we compute the remainders of P with respect to each of the polynomials at the nodes, in a top-down fashion. For instance, the value at α_1 is obtained as

$$P(\alpha_1) = ((P \operatorname{rem} (x-\alpha_1)(x-\alpha_2)(x-\alpha_3)(x-\alpha_4)) \operatorname{rem} (x-\alpha_1)(x-\alpha_2)) \operatorname{rem} (x-\alpha_1).$$

Using Theorem 6, we may use a similar technique for multivariate polynomials and points $\alpha_1, \alpha_2, \alpha_3, \alpha_4 \in \mathbb{K}^n$: we replace each polynomial $(x-\alpha_i)\cdots(x-\alpha_j)$ by the axial basis $\boldsymbol{A}_{(\alpha_i,\ldots,\alpha_j)}$:

$$\begin{tikzcd}[column sep=tiny] & & & \boldsymbol{A}_{(\alpha_1,\alpha_2,\alpha_3,\alpha_4)} \arrow[dll, dash] \arrow[drr, dash] & & & \\ & \boldsymbol{A}_{(\alpha_1,\alpha_2)} \arrow[dl, dash] \arrow[dr, dash] & & & & \boldsymbol{A}_{(\alpha_3,\alpha_4)} \arrow[dl, dash] \arrow[dr, dash] & \\ \boldsymbol{A}_{(\alpha_1)} & & \boldsymbol{A}_{(\alpha_2)} & & \boldsymbol{A}_{(\alpha_3)} & & \boldsymbol{A}_{(\alpha_4)} \end{tikzcd} \tag{5}$$

We may then compute the evaluation of P at α_1 using

$$P(\alpha_1) = ((P \operatorname{rem} \boldsymbol{A}_{(\alpha_1,\alpha_2,\alpha_3,\alpha_4)}) \operatorname{rem} \boldsymbol{A}_{(\alpha_1,\alpha_2)}) \operatorname{rem} \boldsymbol{A}_{(\alpha_1)}.$$

We will call (5) an *evaluation tree.*

5.1. Evaluation trees and multi-point evaluation

In order to specify our general algorithms for multi-point evaluation and the computation of evaluation trees, it is convenient to introduce a few more notations. Given a vector $\boldsymbol{v} = (v_1, \ldots, v_d)$, we will write

$$\begin{aligned} \boldsymbol{v}_{\triangleleft} &:= (v_1, \ldots, v_{\lfloor d/2 \rfloor}), \\ \boldsymbol{v}_{\triangleright} &:= (v_{\lfloor d/2 \rfloor + 1}, \ldots, v_d), \end{aligned}$$

where $\lfloor a \rfloor$ represents the largest integer smaller or equal to a. The least integer larger or equal to a is written $\lceil a \rceil$.

Given vectors $\boldsymbol{u} = (u_1, \ldots, u_c)$ and $\boldsymbol{v} = (v_1, \ldots, v_d)$, we also write

$$\boldsymbol{u} \bowtie \boldsymbol{v} := (u_1, \ldots, u_c, v_1, \ldots, v_d)$$

for their concatenation. Given $\boldsymbol{\alpha} \in (\mathbb{K}^n)^d$, we may use the following recursive algorithm to compute the evaluation tree for $\boldsymbol{\alpha}$:

```
Algorithm EvalTree($\boldsymbol{\alpha}$)
Input: a vector of points $\boldsymbol{\alpha} \in (\mathbb{K}^n)^d$.
Output: an evaluation tree for $\boldsymbol{\alpha}$.

Compute the axial basis $\boldsymbol{A}_{\boldsymbol{\alpha}}$ for $I_{\boldsymbol{\alpha}}$.
If $d = 1$, then return a leaf, labeled by $\boldsymbol{A}_{\boldsymbol{\alpha}}$.
Let $T_{\triangleleft} :=$ EvalTree($\boldsymbol{\alpha}_{\triangleleft}$) and $T_{\triangleright} :=$ EvalTree($\boldsymbol{\alpha}_{\triangleright}$).
Return the tree with root labeled by $\boldsymbol{A}_{\alpha}$ and children $T_{\triangleleft}$, $T_{\triangleright}$.
```

We regard the computation of an evaluation tree as a precomputation. Given an evaluation tree T for $\boldsymbol{\alpha}$, we may efficiently evaluate polynomials $P \in \mathbb{K}[\boldsymbol{x}]$ at all points $\alpha_1, \ldots, \alpha_d$ using the following algorithm:

```
Algorithm Eval($P, T$)
Input: a polynomial $P \in \mathbb{K}[\boldsymbol{x}]$ and an evaluation tree $T$ for $\boldsymbol{\alpha} \in (\mathbb{K}^n)^d$.
Output: the vector $P(\boldsymbol{\alpha}) = (P(\alpha_1), \ldots, P(\alpha_d))$.

Let $\boldsymbol{A}_{\boldsymbol{\alpha}}$ be the axial basis attached to the root of $T$.
Let $R := P \operatorname{rem} \boldsymbol{A}_{\boldsymbol{\alpha}}$.
If $d = 1$, then return $(R)$.
Let $T_{\triangleleft}$ and $T_{\triangleright}$ be the two children of the root of $T$.
Return Eval($R, T_{\triangleleft}$) $\bowtie$ Eval($R, T_{\triangleright}$).
```

5.2. Complexity analysis

The algorithms **EvalTree** and **Eval** actually work for arbitrary point vectors $\boldsymbol{\alpha} \in (\mathbb{K}^n)^d$: it suffices to use a general purpose algorithm to compute Gröbner bases for the ideals $I_{\boldsymbol{\alpha}}$, from which we can then extract the required axial bases. We say that $\boldsymbol{\alpha}$ is *hereditarily generic* if $M_{(\alpha_i,\ldots,\alpha_j),\mathfrak{N}_{j+1-i}}$ is invertible for all $1 \leqslant i \leqslant j \leqslant d$. In that case, each of the required axial bases during recursive calls can be computed using Proposition 3.

THEOREM 8. *The algorithms* **EvalTree** *and* **Eval** *are correct. Let $P \in \mathbb{K}[\boldsymbol{x}]$ with $\deg_{x_i} P < \pi_i$ and $\pi_i \geqslant \delta_{d,i}$ for $i = 1, \ldots, n$, and $\pi := \pi_1 \cdots \pi_n$. If $\boldsymbol{\alpha}$ is hereditarily generic, then the running times of* **EvalTree** *and* **Eval** *are respectively bounded by*

$$\begin{aligned} \mathsf{T}_{\mathbf{EvalTree}}(d) &= O(d^{\omega}) \\ \mathsf{T}_{\mathbf{Eval}}(d, \pi_1, \ldots, \pi_n) &= O(\mathsf{M}(d) \log^3 d + \mathsf{M}(\pi) \log^2 \pi), \end{aligned}$$

whenever an element of multiplicative order $\geqslant (n+1)!\, \pi$ is given in $\mathbb{K}$.

Proof. By induction on d, the algorithms **EvalTree** and **Eval** are clearly correct. Using Proposition 3 and linear algebra, we see that the computation of $\boldsymbol{A}_{\boldsymbol{\alpha}}$ in the first step of **EvalTree** can be done in time $C d^{\omega}$, for some constant C. The running time **EvalTree** therefore satisfies

$$\begin{aligned} \mathsf{T}_{\mathbf{EvalTree}}(1) &\leqslant C \\ \mathsf{T}_{\mathbf{EvalTree}}(d) &\leqslant C d^{\omega} + \mathsf{T}_{\mathbf{EvalTree}}(\lfloor d/2 \rfloor) + \mathsf{T}_{\mathbf{EvalTree}}(\lceil d/2 \rceil), \qquad d \geqslant 2. \end{aligned}$$

By induction on d, this yields $\mathsf{T}_{\mathbf{EvalTree}}(d) \leqslant \bar{\mathsf{T}}_{\mathbf{EvalTree}}(d)$, where

$$\begin{aligned} \bar{\mathsf{T}}_{\mathbf{EvalTree}}(1) &= C \\ \bar{\mathsf{T}}_{\mathbf{EvalTree}}(d) &= C d^{\omega} + \bar{\mathsf{T}}_{\mathbf{EvalTree}}(\lfloor d/2 \rfloor) + \bar{\mathsf{T}}_{\mathbf{EvalTree}}(\lceil d/2 \rceil), \qquad d \geqslant 2. \end{aligned}$$

Again using induction on d, we also note that $\bar{\mathsf{T}}_{\mathbf{EvalTree}}$ is increasing. It follows that $\bar{\mathsf{T}}_{\mathbf{EvalTree}}(d) \leqslant \bar{\mathsf{T}}_{\mathbf{EvalTree}}(\bar{d})$ for $\bar{d} = 2^{\lceil \log d / \log 2 \rceil} \leqslant 2d$. Unrolling the equation

$$\begin{aligned} \bar{\mathsf{T}}_{\mathbf{EvalTree}}(1) &= C \\ \bar{\mathsf{T}}_{\mathbf{EvalTree}}(\bar{d}) &= C \bar{d}^{\omega} + 2 \bar{\mathsf{T}}_{\mathbf{EvalTree}}(\bar{d}/2), \qquad \bar{d} \geqslant 2, \end{aligned}$$

we find

$$\bar{\mathsf{T}}_{\mathbf{EvalTree}}(\bar{d}) \leqslant C \bar{d}^{\omega} + 2C \left(\frac{\bar{d}}{2}\right)^{\omega} + 4C \left(\frac{\bar{d}}{4}\right)^{\omega} + \cdots = \frac{C}{1 - 2^{1-\omega}} \bar{d}^{\omega} = O(d^{\omega}).$$

As to the running time $\mathsf{T}_{\mathbf{Eval}}$ of **Eval**, we recall that $|\mathfrak{N}_d| \sim n!\, d$, whence $|\mathfrak{N}_d| \leqslant K d$ for some constant K that depends on n. Theorem 6 also implies the existence of a constant C such that

$$\begin{aligned} \mathsf{T}_{\mathbf{Eval}}(d, \pi_1, \ldots, \pi_n) &\leqslant C \mathsf{M}(\pi) \log^2 \pi + \mathsf{T}_{\mathbf{Eval}}(d, \delta_{d,1}, \ldots, \delta_{d,n}) \\ \mathsf{T}_{\mathbf{Eval}}(1, 1, \ldots, 1) &\leqslant C \mathsf{M}(K) \\ \mathsf{T}_{\mathbf{Eval}}(d, \delta_{d,1}, \ldots, \delta_{d,n}) &\leqslant C \mathsf{M}(K d) \log^2(K d) + \mathsf{T}_{\mathbf{Eval}}(\lfloor d/2 \rfloor, \delta_{d,1}, \ldots, \delta_{d,n}) \\ &\quad + \mathsf{T}_{\mathbf{Eval}}(\lceil d/2 \rceil, \delta_{d,1}, \ldots, \delta_{d,n}), \qquad d \geqslant 2. \end{aligned}$$

Modulo a further increase of C, the first and third relation may be combined such as to replace the third relation by

$$\begin{aligned} \mathsf{T}_{\mathbf{Eval}}(d, \delta_{d,1}, \ldots, \delta_{d,n}) &\leqslant C \mathsf{M}(K d) \log^2(K d) + \mathsf{T}_{\mathbf{Eval}}(\lfloor d/2 \rfloor, \delta_{\lfloor d/2 \rfloor,1}, \ldots, \delta_{\lfloor d/2 \rfloor,n}) \\ &\quad + \mathsf{T}_{\mathbf{Eval}}(\lceil d/2 \rceil, \delta_{\lceil d/2 \rceil,1}, \ldots, \delta_{\lceil d/2 \rceil,n}), \qquad d \geqslant 2. \end{aligned}$$

By unrolling the latter inequality in a similar way as above for powers of two, we obtain $\bar{\mathsf{T}}_{\mathbf{Eval}}(\bar{d}, \delta_{d,1}, \ldots, \delta_{d,n}) \leqslant C\,\mathsf{M}(K\bar{d})\,(\log(K\bar{d})+1)^3$, while using our assumption that $\mathsf{M}(d)/d$ is non-decreasing. □

Remark 9. If $n=2$ and the first coordinates of the evaluation points are pairwise distinct, then precomputations can be handled more efficiently as follows. The annihilating polynomial takes $O(\mathsf{M}(d)\log d)$ operations in $\mathbb{K}$, using the formula

$$\chi(x_1) := \prod_{i=1}^{d} (x_1 - \alpha_{i,1}).$$

With a similar cost we can also interpolate the polynomial $v_2(x_1)$ of degree $<d$ satisfying $\alpha_{i,2} = v_2(\alpha_{i,1})$ for $i=1,\ldots,d$.

Setting $\nu_j := \max(\deg_{x_1} \mathfrak{m}_i : \deg_{x_2} \mathfrak{m}_i = j, i=1,\ldots,n) + 1$ for $j=0,\ldots,\delta_{d,2}-1$, we note that $\nu_0 + \cdots + \nu_{\delta_{d,2}-1} = d$. Determining the coordinates of $\mathfrak{n} \in \mathfrak{M} \setminus \mathfrak{F}$ in the basis $\mathfrak{F}$ of the quotient ring of $I_{\boldsymbol{\alpha}}$ reduces to computing polynomials $g_0, \ldots, g_{\delta_{d,2}-1}$ in $\mathbb{K}[x_1]$ of respective degree less than $\nu_0, \ldots, \nu_{\delta_{d,2}-1}$ and a scalar $c \in \mathbb{K}$ such that

$$g_0(x_1) + g_1 v_2(x_1) + \cdots + g_{\delta_{d,2}-1}(x_1)\, v_2(x_1)^{\delta_{d,2}-1} + c\,\mathfrak{n}(x_1, v_2(x_1)) = 0 \bmod \chi(x_1).$$

Assuming that $|\mathbb{K}| \geqslant 2d^2$, a non-trivial solution can be computed in expected time $O(\delta_{d,2}^{\omega-1}\,\mathsf{M}(d)\log^2 d) = \tilde{O}(d^{(\omega+1)/2})$ using the probabilistic algorithm of Las Vegas type underlying [4, Corollary 1]. Since $M_{\boldsymbol{\alpha},\mathfrak{F}}$ is invertible, the value found for c cannot be zero, and we obtain the solution of (1) in this way.

6. INTERPOLATION

In the univariate case, the technique of remainder trees can also be used to efficiently reconstruct a polynomial $P \in \mathbb{K}[x]$ from its values at d distinct points $\alpha_1, \ldots, \alpha_d$. This basically amounts to reversing the evaluation process, while using the Chinese remainder theorem to reconstruct $P \operatorname{rem} (A_\triangleleft A_\triangleright)$ from $P \operatorname{rem} A_\triangleleft$ and $P \operatorname{rem} A_\triangleright$ for coprime polynomials $A_\triangleleft$ and $A_\triangleright$. For such reconstructions using the Chinese remainder theorem, it is useful to precompute the "cofactors" $C_\triangleleft, C_\triangleright \in \mathbb{K}[x]$ with $C_\triangleleft A_\triangleleft \operatorname{rem} A_\triangleright = 1$, $C_\triangleright A_\triangleright \operatorname{rem} A_\triangleleft = 1$, $\deg C_\triangleleft < \deg A_\triangleright$, and $\deg C_\triangleright < \deg A_\triangleleft$, after which

$$P \operatorname{rem} (A_\triangleleft A_\triangleright) = ((P \operatorname{rem} A_\triangleright)\, C_\triangleleft A_\triangleleft + (P \operatorname{rem} A_\triangleleft)\, C_\triangleright A_\triangleright) \operatorname{rem} (A_\triangleleft A_\triangleright).$$

These cofactors are typically stored in an upgraded version of the remainder tree.

In our multivariate setting, a similar idea works, modulo some additional precautions when computing the cofactors. The cofactors are most easily constructed *via* their evaluations. Indeed, consider a tuple of points $\boldsymbol{\alpha} \in (\mathbb{K}^n)^d$ with $d \geqslant 2$ and its decomposition $\boldsymbol{\alpha} = \boldsymbol{\alpha}_\triangleleft \bowtie \boldsymbol{\alpha}_\triangleright$. Then the first element $A_\triangleleft := A_{\boldsymbol{\alpha}_\triangleleft,1}$ of the axial basis of $I_{\boldsymbol{\alpha}_\triangleleft}$ certainly vanishes at $\boldsymbol{\alpha}_\triangleleft$ and we wish to let it play the same role as $A_\triangleleft$ above. The corresponding cofactor $C_\triangleleft$ should satisfy $(C_\triangleleft A_\triangleleft)(\boldsymbol{\alpha}_\triangleright) = 1$, so we may compute it through interpolation from its evaluation $A_\triangleleft(\boldsymbol{\alpha}_\triangleright)^{-1}$ at $\boldsymbol{\alpha}_\triangleright$. However, this requires $A_\triangleleft$ not to vanish at any of the entries of $\boldsymbol{\alpha}_\triangleright$. Now the set of tuples of points $\boldsymbol{\alpha} \in (\mathbb{K}^n)^d$ for which one of the entries of $A_\triangleleft(\boldsymbol{\alpha}_\triangleright)$ vanishes forms a Zariski closed subset of dimension $n\,d-1$; for a "generic" tuple of points, both $A_\triangleleft(\boldsymbol{\alpha}_\triangleright)$ and $A_\triangleright(\boldsymbol{\alpha}_\triangleleft)$ (where $A_\triangleright := A_{\boldsymbol{\alpha}_\triangleright,1}$) are therefore invertible. Given $P \in \mathbb{K}[\boldsymbol{x}]$, this allows us to reconstruct $P \operatorname{rem} \boldsymbol{A}_{\boldsymbol{\alpha}}$ from $P \operatorname{rem} \boldsymbol{A}_{\boldsymbol{\alpha}_\triangleright}$ and $P \operatorname{rem} \boldsymbol{A}_{\boldsymbol{\alpha}_\triangleleft}$ using

$$P \operatorname{rem} \boldsymbol{A}_{\boldsymbol{\alpha}} = ((P \operatorname{rem} \boldsymbol{A}_{\boldsymbol{\alpha}_\triangleright})\, C_\triangleleft A_\triangleleft + (P \operatorname{rem} \boldsymbol{A}_{\boldsymbol{\alpha}_\triangleleft})\, C_\triangleright A_\triangleright) \operatorname{rem} \boldsymbol{A}_{\boldsymbol{\alpha}}.$$

More generally, we define $\boldsymbol{\alpha}$ to be *super-generic* if it is hereditarily generic and, for all $1 \leqslant i \leqslant j \leqslant d$ and $k \in \{1, \ldots, n\} \setminus \{i, \ldots, j\}$, we have $A_{(\alpha_i, \ldots, \alpha_j), 1}(\alpha_k) \neq 0$.

6.1. Interpolation trees and interpolation

Let us now detail the interpolation algorithm that was summarized and explained above. Similarly to the case of multi-point evaluation, it relies on the construction of an *interpolation tree*, which contains the required cofactors. Contrary to before, the construction of this tree recursively involves interpolations (in order to reconstruct the cofactors from their evaluations).

```
**Algorithm IpolTree($T$)**
**Input:** an evaluation tree $T$ for a super-generic vector $\boldsymbol{\alpha} \in (\mathbb{K}^n)^d$.
**Output:** an interpolation tree $U$ for $\boldsymbol{\alpha}$.

If $d = 1$, then return a leaf.
Let $\boldsymbol{A}_{\boldsymbol{\alpha}}$ be the axial Gröbner basis attached to the root of $T$.
Recursively apply the algorithm to the children $T_\triangleleft$, $T_\triangleright$ of $T$, yielding $U_\triangleleft$, $U_\triangleright$.
Let $A_\triangleleft$, $A_\triangleright$ be the first entries of the axial bases associated to the roots of $T_\triangleleft$, $T_\triangleright$.
Compute $E_\triangleleft := \mathbf{Eval}(A_\triangleleft, T_\triangleright)$ and $E_\triangleright := \mathbf{Eval}(A_\triangleright, T_\triangleleft)$.
Compute $C_\triangleleft := \mathbf{Ipol}(E_\triangleleft^{-1}, U_\triangleright)$ and $C_\triangleright := \mathbf{Ipol}(E_\triangleright^{-1}, U_\triangleleft)$.
Return the tree with root labeled by $(\boldsymbol{A}_{\boldsymbol{\alpha}}, C_\triangleleft, C_\triangleright)$ and children $U_\triangleleft$, $U_\triangleright$.
```

```
**Algorithm Ipol($v, U$)**
**Input:** $v \in \mathbb{K}^d$ and the interpolation tree $U$ for $\boldsymbol{\alpha} \in (\mathbb{K}^n)^d$.
**Output:** $P \in \mathrm{Red}_d$ with $P(\boldsymbol{\alpha}) = v$.

If $d = 1$, then return $v_1$.
Let $A_\triangleleft$, $A_\triangleright$ be the first entries of the axial bases associated to the roots of $T_\triangleleft$, $T_\triangleright$.
Let $U_\triangleleft$, $U_\triangleright$ be the children of $U$.
Let $(\boldsymbol{A}_{\boldsymbol{\alpha}}, C_\triangleleft, C_\triangleright)$ be the label of the root of $U$.
Let $P_\triangleleft := \mathbf{Ipol}(v_\triangleleft, U_\triangleleft)$ and $P_\triangleright := \mathbf{Ipol}(v_\triangleright, U_\triangleright)$.
Return $((P_\triangleright C_\triangleleft \operatorname{rem} \boldsymbol{A}_{\boldsymbol{\alpha}_\triangleright}) A_\triangleleft + (P_\triangleleft C_\triangleright \operatorname{rem} \boldsymbol{A}_{\boldsymbol{\alpha}_\triangleleft}) A_\triangleright) \operatorname{rem} \boldsymbol{A}_{\boldsymbol{\alpha}}$.
```

6.2. Complexity analysis

THEOREM 10. *The algorithms* **IpolTree** *and* **Ipol** *are correct. If $\boldsymbol{\alpha}$ is super generic, then the running times of* **IpolTree** *and* **Ipol** *are respectively bounded by*

$$\begin{aligned} \mathsf{T}_{\mathbf{IpolTree}}(d) &= O(\mathsf{M}(d) \log^4 d) \\ \mathsf{T}_{\mathbf{Ipol}}(d) &= O(\mathsf{M}(d) \log^3 d), \end{aligned}$$

whenever an element of multiplicative order $\geqslant (n+1)! 2^n \delta$ is given in $\mathbb{K}$, where $\delta := \delta_{d,1} \cdots \delta_{d,n}$.

Proof. By induction on d, the algorithms **IpolTree** and **Ipol** are clearly correct. Let us start with the complexity bound for **Ipol**. We have $\deg_{x_i}(P_\triangleright C_\triangleleft) < 2\delta_{d,i}$, $\deg_{x_i}(P_\triangleleft C_\triangleright) < 2\delta_{d,i}$, $\deg_{x_i}((P_\triangleright C_\triangleleft \operatorname{rem} \boldsymbol{A}_{\boldsymbol{\alpha}_\triangleright}) A_\triangleleft) < 2\delta_{d,i}$, and $\deg_{x_i}((P_\triangleleft C_\triangleright \operatorname{rem} \boldsymbol{A}_{\boldsymbol{\alpha}_\triangleleft}) A_\triangleright)$, for $i = 1, \ldots, n$, where $(2\delta_{d,1}) \cdots (2\delta_{d,n}) = O(2^n n! d) = O(d)$. Theorem 6 therefore implies the existence of constants C and K with

$$\begin{aligned} \mathsf{T}_{\mathbf{Ipol}}(1) &\leqslant C \mathsf{M}(K) \\ \mathsf{T}_{\mathbf{Ipol}}(d) &\leqslant C \mathsf{M}(K d) \log^2(K d) + \mathsf{T}_{\mathbf{Ipol}}(\lfloor d/2 \rfloor) + \mathsf{T}_{\mathbf{Ipol}}(\lceil d/2 \rceil), \qquad d \geqslant 2. \end{aligned}$$

By unrolling the latter inequality in a similar way as in the proof of Theorem 8 for powers of two, we obtain $\bar{\mathsf{T}}_{\mathbf{Ipol}}(\bar{d}) \leqslant C\,\mathsf{M}(K\bar{d})\,(\log(K\bar{d})+1)^3$, while using our assumption that $\mathsf{M}(d)/d$ non-decreasing.

As to the construction of the interpolation tree, Theorem 6, together with the bound for $\mathsf{T}_{\mathbf{Eval}}$ of Theorem 8, and $\mathsf{T}_{\mathbf{Ipol}}$ imply the existence of other constants C and K with

$$\begin{aligned} \mathsf{T}_{\mathbf{IpolTree}}(1) &\leqslant C\,\mathsf{M}(K) \\ \mathsf{T}_{\mathbf{IpolTree}}(d) &\leqslant C\,\mathsf{M}(Kd)\log^3(Kd) + \mathsf{T}_{\mathbf{IpolTree}}(\lfloor d/2 \rfloor) + \mathsf{T}_{\mathbf{IpolTree}}(\lceil d/2 \rceil), \qquad d \geqslant 2. \end{aligned}$$

Unrolling the latter inequality yields $\bar{\mathsf{T}}_{\mathbf{IpolTree}}(\bar{d}) \leqslant C\,\mathsf{M}(K\bar{d})\,(\log(K\bar{d})+1)^4$. □

BIBLIOGRAPHY

[1] J. Abbott, A. Bigatti, M. Kreuzer, and L. Robbiano. Computing ideals of points. *J. Symbolic Comput.*, 30(4):341–356, 2000.

[2] S. Abelard, A. Couvreur, and G. Lecerf. Sub-quadratic time for Riemann–Roch spaces. The case of smooth divisors over nodal plane projective curves. Technical Report, HAL, 2020. `https://hal.archives-ouvertes.fr/hal-02477371`.

[3] A. Borodin and R. T. Moenck. Fast modular transforms. *J. Comput. System Sci.*, 8:366–386, 1974.

[4] A. Bostan, C.-P. Jeannerod, and É. Schost. Solving structured linear systems with large displacement rank. *Theor. Comput. Sci.*, 407(1):155–181, 2008.

[5] A. Bostan, G. Lecerf, and É. Schost. Tellegen's principle into practice. In *Proceedings of the 2003 International Symposium on Symbolic and Algebraic Computation*, ISSAC '03, pages 37–44. New York, NY, USA, 2003. ACM.

[6] D. G. Cantor and E. Kaltofen. On fast multiplication of polynomials over arbitrary algebras. *Acta Inform.*, 28:693–701, 1991.

[7] M. F. I. Chowdhury, C. Jeannerod, V. Neiger, É. Schost, and G. Villard. Faster algorithms for multivariate interpolation with multiplicities and simultaneous polynomial approximations. *IEEE Trans. Inf. Theory*, 61(5):2370–2387, 2015.

[8] J.-C. Faugère, P. Gaudry, L. Huot, and G. Renault. Sub-cubic change of ordering for Gröbner basis: a probabilistic approach. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*, ISSAC '14, pages 170–177. New York, NY, USA, 2014. ACM.

[9] C. M. Fiduccia. Polynomial evaluation via the division algorithm: the fast Fourier transform revisited. In *Proceedings of the Fourth Annual ACM Symposium on Theory of Computing*, STOC '72, pages 88–93. New York, NY, USA, 1972. ACM.

[10] M. Gasca and T. Sauer. Polynomial interpolation in several variables. *Adv. Comput. Math.*, 12(4):377, 2000.

[11] D. Harvey and J. van der Hoeven. Faster polynomial multiplication over finite fields using cyclotomic coefficient rings. *J. Complexity*, 54:101404, 2019.

[12] J. van der Hoeven. Faster relaxed multiplication. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*, ISSAC '14, pages 405–412. New York, NY, USA, 2014. ACM.

[13] J. van der Hoeven. Faster Chinese remaindering. Technical Report, CNRS & École polytechnique, 2016. `http://hal.archives-ouvertes.fr/hal-01403810`.

[14] J. van der Hoeven. On the complexity of multivariate polynomial division. In I. S. Kotsireas and E. Martínez-Moro, editors, *Applications of Computer Algebra. Kalamata, Greece, July 20–23, 2015*, volume 198 of *Springer Proceedings in Mathematics & Statistics*, pages 447–458. Cham, 2017. Springer International Publishing.

[15] J. van der Hoeven and G. Lecerf. On the bit-complexity of sparse polynomial multiplication. *J. Symbolic Comput.*, 50:227–254, 2013.

[16] J. van der Hoeven and G. Lecerf. On the complexity exponent of polynomial system solving. Technical Report, HAL, 2018. `http://hal.archives-ouvertes.fr/hal-01848572`.

[17] J. van der Hoeven and G. Lecerf. Fast multivariate multi-point evaluation revisited. *J. Complexity*, 56:101405, 2020.

[18] J. van der Hoeven et al. GNU TeXmacs. `http://www.texmacs.org`, 1998.

[19] K. S. Kedlaya and C. Umans. Fast polynomial factorization and modular composition. *SIAM J.Comput.*, 40(6):1767–1802, 2011.

[20] D. Le Brigand and J.-J. Risler. Algorithme de Brill–Noether et codes de Goppa. *Bulletin de la société mathématique de France*, 116(2):231–253, 1988.

[21] F. Le Gall. Powers of tensors and fast matrix multiplication. In K. Nabeshima, editor, *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*, ISSAC '14, pages 296–303. New York, NY, USA, 2014. ACM.

[22] M. G. Marinari, H. M. Möller, and T. Mora. Gröbner bases of ideals defined by functionals with an application to ideals of projective points. *Appl. Algebra Eng. Commun. Comput.*, 4(2):103–145, 1993.

[23] R. T. Moenck and A. Borodin. Fast modular transforms via division. In *13th Annual Symposium on Switching and Automata Theory*, pages 90–96. USA, 1972. IEEE.

[24] H. M. Möller and B. Buchberger. The construction of multivariate polynomials with preassigned zeros. In J. Calmet, editor, *Computer Algebra. EUROCAM '82, European Computer Algebra Conference. Marseille, France 5–7 April 1982*, volume 144 of *Lect. Notes Comput. Sci.*, pages 24–31. Berlin, Heidelberg, 1982. Springer Berlin Heidelberg.

[25] V. Neiger. Fast computation of shifted Popov forms of polynomial matrices via systems of modular polynomial equations. In *Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation*, ISSAC '16, pages 365–372. New York, NY, USA, 2016. ACM.

[26] M. Nüsken and M. Ziegler. Fast multipoint evaluation of bivariate polynomials. In S. Albers and T. Radzik, editors, *Algorithms – ESA 2004. 12th Annual European Symposium, Bergen, Norway, September 14-17, 2004*, volume 3221 of *Lect. Notes Comput. Sci.*, pages 544–555. Springer Berlin Heidelberg, 2004.

[27] D. S. Roche. What can (and can't) we do with sparse polynomials? In C. Arreche, editor, *ISSAC '18: Proceedings of the 2018 ACM International Symposium on Symbolic and Algebraic Computation*, pages 25–30. ACM Press, 2018.

[28] A. Schönhage. Schnelle Multiplikation von Polynomen über Körpern der Charakteristik 2. *Acta Infor.*, 7:395–398, 1977.

[29] A. Schönhage and V. Strassen. Schnelle Multiplikation großer Zahlen. *Computing*, 7:281–292, 1971.

[30] V. Shoup. New algorithms for finding irreducible polynomials over finite fields. *Math. Comp.*, 54(189):435–447, 1990.