Electronic Colloquium on Computational Complexity, Report No. 33 (2023)

Fast Numerical Multivariate Multipoint Evaluation

Sumanta Ghosh * Prahladh Harsha† Simao Herdade‡ Mrinal Kumar†
Ramprasad Saptharishi†

Abstract

We design nearly-linear time numerical algorithms for the problem of multivariate multipoint evaluation over the fields of rational, real and complex numbers. We consider both *exact* and *approximate* versions of the algorithm. The input to the algorithms are (1) coefficients of an m-variate polynomial f with degree d in each variable, and (2) points $\mathbf{a}_1, \ldots, \mathbf{a}_N$ each of whose coordinate has value bounded by one and bit-complexity s.

Approximate version: Given additionally an accuracy parameter t, the algorithm computes rational numbers $\beta_1, \ldots, \beta_N$ such that $|f(\mathbf{a}_i) - \beta_i| \leq 1/2^t$ for all i, and has a running time of $((Nm + d^m)(s + t))^{1+o(1)}$ for all m and all sufficiently large d.

Exact version (when over rationals): Given additionally a bound c on the bit-complexity of all evaluations, the algorithm computes the rational numbers $f(\mathbf{a}_1), \ldots, f(\mathbf{a}_N)$, in time $((Nm + d^m)(s + c))^{1+o(1)}$ for all m and all sufficiently large d. .

Our results also naturally extend to the case when the input is over the field of real or complex numbers under an appropriate standard model of representation of field elements in such fields.

Prior to this work, a nearly-linear time algorithm for multivariate multipoint evaluation (exact or approximate) over any infinite field appears to be known only for the case of univariate polynomials, and was discovered in a recent work of Moroz [Mor21]. In this work, we extend this result from the univariate to the multivariate setting. However, our algorithm is based on ideas that seem to be conceptually different from those of Moroz [Mor21] and crucially relies on a recent algorithm of Bhargava, Ghosh, Guo, Kumar & Umans [BGGKU22] for multivariate multipoint evaluation over finite fields, and known efficient algorithms for the problems of rational number reconstruction and fast Chinese remaindering in computational number theory.

*California Institute of Technology, Pasadena, USA. besusumanta@gmail.com

†Tata Institute of Fundamental Research, Mumbai, India. {prahladh, mrinal, ramprasad}@tifr.res.in. Research supported by the Department of Atomic Energy, Government of India, under project 12-R&D-TFR-5.01-0500. Research of second author partially supported by Google India Research Award.

‡Yahoo Research, San Francisco, USA. simaoherdade@gmail.com

git info: (None) , ((None))

ISSN 1433-8092

1 Introduction

In this paper, we study the problem of designing fast algorithms for the following natural computational problem.

> Given an m variate polynomial f of degree less than d in each variable over an underlying field $\mathbb{F}$ as a list of coefficients, and (arbitrary) evaluation points $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N \in \mathbb{F}^m$, output $f(\mathbf{a}_i)$ for every i.

This computational task is referred to as *Multivariate Multipoint Evaluation (MME)* in literature and fast algorithms for MME are of fundamental interest in computational algebra, not only due to the evident natural appeal of the problem but also due to potential applications of MME as an important subroutine for algorithms for many other algebraic problems (see [KU11] for a detailed discussion on these applications).

The input for MME is specified by $(d^m + Nm)$ field elements, or alternatively $(d^m + Nm) \cdot s$ bits, where s is an upper bound on the bit complexity of any field constant in the input. For finite fields, s can be taken to be $\log |\mathbb{F}|$. Clearly, there is an algorithm for this problem that takes roughly $(d^m \cdot Nm)^{1+o(1)}$ many field operations or about $(d^m \cdot Nm \cdot s)^{1+o(1)}$ bit operations: we iteratively evaluate the polynomial one input point at a time. Obtaining significantly faster and ideally *nearly-linear*[1] time algorithms for MME is the main question motivating this work. Here the time complexity of an algorithm could be measured either in terms of the number of field operations (in case the algorithm is "algebraic[2]" in the sense that only uses field operations over the underlying field, e.g. like the trivial algorithm outlined above) or the number of bit operations.

1.1 Prior work

Before describing the precise problem studied in this work and our main results, we start with a brief discussion of the current state of art of algorithms for MME. While the results in this paper are over infinite fields like reals, rationals and complex numbers, we begin our discussion of prior work on MME by recalling the state of affairs over finite fields.

1.1.1 Multipoint evaluation over finite fields

Multipoint evaluation of polynomials is a non-trivial problem even for the case of univariate polynomials, and a non-trivial algorithm is unclear even for this case over any (sufficiently large) field. When the set of input points have additional structure, for instance, they are all roots of unity of some order over the underlying field, the Fast Fourier Transform (FFT) gives us a nearly-linear time algorithm for this problem. However, it is not immediately clear whether ideas based on FFT can be easily extended to the case of arbitrary evaluation points.

[1] We say that an algorithm has time complexity nearly-linear in the input size if for all sufficiently large n, the algorithms runs on inputs of size n in time $n^{1+o(1)}$.

[2] Algorithms for MME that only need arithmetic over the underlying field in their execution, or in other words can be modelled as an arithmetic circuit over the underlying field are referred to as algebraic.

In a beautiful work in 1974, Borodin and Moenck [BM74] designed a significantly faster algorithm for univariate multipoint evaluation by building on FFT and a fast algorithm for division with remainder for univariate polynomials. The algorithm of Borodin and Moenck worked over all fields and was algebraic, in the sense mentioned earlier, the number of field operations needed by the algorithm was $(N+d)^{1+o(1)}$, nearly-linear in the number of field elements in the input.

Extending these fast algorithms for multipoint evaluation from the univariate to the multivariate case proved to be quite challenging, even for the bivariate case. Nüsken and Ziegler [NZ04] gave a non-trivially fast algorithm for this problem over all fields, although the precise time complexity of their algorithm was not nearly linear in the input size. The state of art for this problem saw a significant improvement in the works of Umans [Uma08] and Kedlaya & Umans [KU08] (see also [KU11]) who gave fast algorithms for MME for the case when the number of variables m is significantly smaller than the degree parameter d, i.e. $m = d^{o(1)}$, over fields of small characteristic and all finite fields respectively.

This case of large number of variables was addressed recently in works of Bhargava, Ghosh, Kumar & Mohapatra [BGKM22] and Bhargava, Ghosh, Guo, Kumar & Umans [BGGKU22] who gave fast[3] algorithms for MME over fields of small characteristic and over all finite fields respectively, for all sufficiently large m, d.

We also note that the algorithms of Kedlaya & Umans [KU08] and those of Bhargava, Ghosh, Guo, Kumar & Umans [BGGKU22] for MME over all finite fields are not algebraic, and in particular rely on bit access to the input field elements and bit operations on them. This is in contrast to the algorithms of Umans [Uma08] and Bhargava, Ghosh, Kumar & Mohapatra [BGKM22] for MME over finite fields of small characteristic that are algebraic in nature. Designing algebraic algorithms for MME over all finite fields continues to be a very interesting open problem in this line of research. ‘

1.1.2 Multipoint evaluation over infinite fields

As we shall see, our understanding of the problem here is rather limited compared to that over finite fields. However, before moving on to the results, we first discuss some subtleties with the definition of this problem itself over infinite fields.

Field operations vs bit complexity: Field arithmetic over finite fields preserves the worst case bit complexity of the constants generated, but this is not the case over infinite fields. This increase in bit-complexity in intermediate computations leads to some issues that we discuss next.

The first issue is that even the bit complexity of the output may not be nearly-linear in the input bit complexity, thereby ruling out any hope of having an algorithm with time complexity nearly-linear in the bit complexity of the input. The second issue is that even for inputs where the

[3]Strictly speaking, these algorithms do not run in nearly-linear time, since the running time has $(\log |\mathbb{F}|)^c$ factor where c is a fixed constant that can be greater than one. However, the dependence of the running time on the term $(d^m + Nm)$ is nearly-linear.

bit complexity of the input field elements and the output field elements are promised to be small, it might be the case that in some intermediate stage of its run, an algorithm for MME generates field elements of significantly large bit complexity. For instance, the classical algorithm of Borodin & Moenck for univariate multipoint evaluation has near linear complexity in terms of the number of field operations, but it is not clear if the bit complexity of the algorithm is also nearly-linear in the input and output bit complexities.

The input and output model: «SG: PH had a comment about decimal vs bit. Not sure what to do. » For fields such as real or complex numbers, we need to specify a model for providing the inputs which potentially require infinite precision. The standard model used in numerical algorithms is via black-boxes that we refer to as *approximation oracles* (formally defined in Definition 2.7). Informally an approximation oracle for a real number $\alpha \in (-1,1)$ provides, for every $k \in \mathbb{N}$, access to the first k bits of α after the decimal, and its sign in time $\tilde{O}(k)$ (for complex numbers, we will assume the real and imaginary parts are provided via such oracles).

For the output, we could either ask to compute the evaluations to the required precision, or compute the evaluations exactly when, say, in the case of rational numbers. In this paper, we consider both versions of these problems.

Note that computing a real number $\alpha \in (-1,1)$ within a given error $\varepsilon < 1$ is essentially the same as computing the most significant $\Omega(\log 1/\varepsilon)$ bits of the output correctly. In this sense, $O(\log 1/\varepsilon)$ provides a natural upper bound on the bit complexity of the output for an instance of approximate multipoint evaluation. Perhaps a bit surprisingly, we did not know an algorithm for multipoint evaluation with bit complexity nearly-linear in input size and $(\log 1/\varepsilon)$ even for the setting of univariate polynomials till very recently. This is in contrast to the result of Borodin & Moenck [BM74] that obtains an upper bound on the number of field operations (but not the number of bit operations) for (exact) univariate multipoint evaluation over all fields.

In a beautiful recent work, Moroz [Mor21] designed such an algorithm for the approximation version of univariate multipoint evaluation. Formally, he proved the following theorem.

Theorem 1.1 (Moroz **[?]**). *There is a deterministic algorithm that takes as input a univariate polynomial $f(x) = \sum_{i=0}^{d} f_i x^i \in \mathbb{C}[x]$ as a list of complex coefficients, with $(|f|_1 := \sum_{i=0}^{d} |f_i| \leq 2^\tau)$ and inputs $a_1, a_2, \ldots, a_d \in \mathbb{C}$ of absolute value less than one, and outputs $\beta_1, \beta_2, \ldots, \beta_d \in \mathbb{C}$ such that for every i,*

$$|f(a_i) - \beta_i| \leq |f|_1 \cdot 2^{-t},$$

and has bit complexity at most $\tilde{O}(d(\tau + t))$.

As our main result in this paper, we prove a generalization of Theorem 1.1 to the multivariate setting.

1.2 Our results

Before stating our results, we formally define the problems that we study. The first question of approximate-MME is essentially the generalization of the univariate version of the problem studied by Moroz [Mor21]. For convenience, we state the problem for the fields of rational and real numbers, but they extend in a straightforward manner to complex numbers and other natural subfields of it.

Problem 1.2 (Approximate multivariate multipoint evaluation (approximate-MME))**.** *We are given as input an m-variate polynomial $f \in \mathbb{R}[\mathbf{x}]$ of degree at most $(d-1)$ in each variable as a list of coefficients, points $\mathbf{a}_1, \ldots, \mathbf{a}_N \in \mathbb{R}^m$, and an accuracy parameter $t \in \mathbb{N}$. Here every field element is assumed to be in $(-1, 1)$ and is given via an approximation oracle.*

Compute rational numbers $\beta_1, \ldots, \beta_N$ such that $|f(\mathbf{a}_i) - \beta_i| < 1/2^t$ for all $i \in [N]$.

We also study the following variant of MME in the paper.

Problem 1.3 (Exact multivariate multipoint evaluation (exact-MME))**.** *We are given as input an m-variate polynomial $f \in \mathbb{Q}[\mathbf{x}]$ of degree at most $(d-1)$ in each variable as a list of coefficients, points $\mathbf{a}_1, \ldots, \mathbf{a}_N \in \mathbb{Q}^m$ and an integer parameter $s > 0$, such that all rational numbers in the input and* output *are expressible in the form p/q for integers p, q with $|p|, |q| < 2^s$ and every rational number in the input has absolute value less than one.*

Compute $f(\mathbf{a}_1), \ldots, f(\mathbf{a}_N)$.

The restriction that the absolute value of all constants is at most one requires a short discussion. The restriction on the coefficients of f is without loss of generality (by scaling) but the restriction on the coordinates of points is *not* without loss of generality, but is nevertheless well-motived. See Remark 5.1 for details.

Our main result is fast algorithms for Problem 1.2 and Problem 1.3 for all sufficiently large d.

Theorem 1.4 (Approximate multipoint evaluation - informal)**.** *There is a deterministic algorithm for approximate-MME (Problem 1.2) that runs in time*

$$((Nm + d^m)t)^{1+o(1)} ,$$

for all sufficiently large d, t and all m.

Theorem 1.5 (Exact multipoint evaluation - informal)**.** *There is a deterministic algorithm for exact-MME over rational numbers (Problem 1.3) that runs in time*

$$((Nm + d^m)s)^{1+o(1)}$$

for all sufficiently large d, s and all m.

Theorem 1.4 is a generalization (by scaling coefficients) of Theorem 1.1 of Moroz in the sense that it handles an arbitrarily large number of variables. Interestingly, our proof is *not* an extension of the proof of Theorem 1.1 to larger number of variables. It relies on a different set of ideas and appears to be conceptually different from the proof of Moroz [Mor21]. Moroz's algorithm relies on geometric ideas, and does not involve any modular arithmetic, whereas ours crucially relies on various reductions from an instance of MME (approximate or exact) over rational, real or complex numbers to instances of MME over finite fields. In fact, a generalization of Moroz's univariate algorithm to higher dimensions is not immediately clear to us, and would be interesting to understand.

As discussed in the introduction, while measuring the complexity of algorithms for MME over the field of rational numbers in terms the number of bit operations, the dependence of the running time on the bit complexity of the output, as in Theorem 1.5 is quite natural and essentially unavoidable. However, the fact that Theorem 1.5 takes the bit complexity of the output as a part of its input does not seem very natural and desirable. It would be very interesting to have an algorithm for exact-MME over rationals that does not need a bound on the output complexity as a part of the input, but runs in time nearly-linear in the input and output bit complexity.

1.3 Overview of the proofs

In this section, we outline the main ideas in the proofs of Theorem 1.4 and Theorem 1.5. For this discussion, we assume for simplicity that the input is over the field of rational numbers, and the field constants in the input are given to us exactly. The ideas here generalize to the setting of real inputs (for approximate-MME) by some clean and simple properties of approximation oracles.

1.3.1 A naïve first attempt

We start by setting up some notation. Let $f \in \mathbb{Q}[\mathbf{x}]$ be an m variate polynomial of degree at most $(d-1)$ in each variable and let $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N \in \mathbb{Q}^m$ be the input points of interest. For now, let us assume that our goal is to output the value of f on each $\mathbf{a}_i$ exactly. We are also given the positive integer t such that the numerator and the denominator of each of the field constants in the input, and the output are at most 2^t.

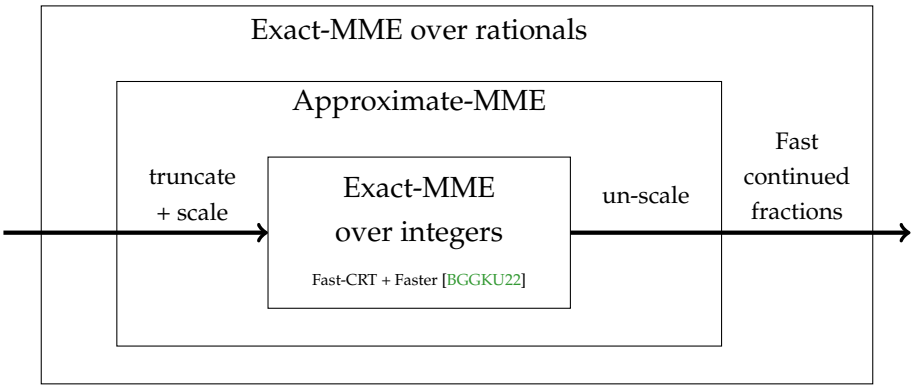
From the prior work of Bhargava, Ghosh, Guo, Kumar and Umans [**?**] we have fast algorithms for MME over all finite fields. Therefore, a natural strategy for solving MME over rational numbers is to somehow reduce the problem over rationals to instances of the problem over finite fields, and use the known algorithms for this problem over finite fields to solve these instances. A first step towards such a reduction would be to clear all the denominators in the input instance by taking their LCM and obtain an instance of MME over the ring of integers, and then work modulo a large enough prime (or several small enough primes if needed for efficiency reasons), thereby reducing to instances of MME over finite fields. However, this seemingly natural approach runs into fundamental issues even for the simpler setting where each evaluation point $\mathbf{a}_i$ has integer

coordinates, and the only rational numbers appear in the coefficients of the polynomial f. We now briefly elaborate on this issue.

Let us consider an input instance where every denominator in the coefficient vector of f is a distinct prime. For instance, we can get such an instance where each of the first d^m primes appears as a denominator of some coefficient of f. Note that the input bit complexity parameter t needs to be at most $\text{poly}(\log d, m)$ for this case. Since the length of this coefficient vector is d^m, the LCM of these denominators is a natural number that is at least as large as the product of the first d^m primes, which is at least 2^{d^m}, and hence has bit complexity at least d^m. Thus, if we clear out the denominators of the coefficients of f to obtain a polynomial $\hat{f}$ with integer coefficients, each of the coefficients of $\hat{f}$ can have bit complexity as large as d^m. In this case, the total bit complexity of the coefficient vector of $\hat{f}$ is at least d^{2m}, which is roughly quadratic in the original input size, and thus, any algorithm obtained via this approach will have prohibitively large time complexity.

In both our algorithms for approximate-MME and exact-MME, we indeed crucially rely on the algorithms for MME over finite fields due to Bhargava et al [BGGKU22]. However, this reduction is somewhat delicate and needs some care. On the way we rely on some well known tools from computational algebra and number theory, like fast Chinese remaindering, fast rational reconstruction, Kronecker and inverse Kronecker maps. Perhaps a bit surprisingly, our algorithm for exact-MME uses the algorithm for approximate-MME as a subroutine.

Figure 1: Overview of reductions



We now give an overview of the main ideas in these algorithms. We start with a very simple algorithm for exact-MME for the special case of integer inputs that serves as a crucial subroutine for the algorithm for approximate-MME.

1.3.2 Algorithm for exact-MME over integers

For this problem, all the field elements in the input are integers and the absolute value of each of these input field elements and those in the output is at most 2^s for a given parameter s.

The algorithm for MME for this case simply does this computation by working modulo a large

enough prime (based on the given input and output complexities), thereby giving us a reduction from the problem over integers to that over a large enough finite field. At this point, we essentially invoke the algorithm of Bhargava et al for MME over finite fields to solve this problem. One subtlety here is that as stated in their paper [BGGKU22], the algorithm does not quite run in nearly-linear time due to two factors. The first issue is that the running time has a $\text{poly}(\log |\mathbb{F}|)$ term, where the degree of $\text{poly}()$ term can be strictly larger than one. The other issue is that even in terms of the dependence on d, m, their algorithm is nearly-linear time only when m is growing. So, for constant m, we cannot directly invoke the algorithm in [BGGKU22] for our applications.

We get around both these issues using some simple ideas. To address the issue of a constant number of variables, we artificially increase the number variables, while reducing the individual degree bound by applying an inverse-Kronecker map to the polynomial. Then, to deal with the issue of dependence of the running time on the field size, we first do a lift to integers and a Chinese remaindering to reduce this problem to many such instances of MME over smaller finite fields. This is essentially the same as the reduction used by Kedlaya & Umans in [KU11]. To keep the running time nearly-linear, we do the Chinese remaindering using the well known nearly-linear time algorithms. The details can be found in Section 3 and Section 4.

1.3.3 Algorithm for approximate-MME

Recall that for approximate-MME, we do not need to compute the value of the polynomial on the input points exactly, but only require the outputs to be within some error of the actual evaluations. For simplicity, let us assume that the input polynomial and the evaluation points are all rational numbers, and are given exactly. As alluded to earlier in this section, it seems difficult to simply clear the denominators (via an LCM) and reduce to the integer case since there are instances, like when the denominators are all distinct primes, where this process prohibitively blows up the size of the coefficients. However, working with approximations gives us the necessary wiggle room to make something close to this idea work.

As the first step of the algorithm, we approximate all the field constants, the coefficients of the given polynomial as well as the coordinates of the input points by truncating their decimal representation to k bits after decimal (for some sufficiently large k to be chosen carefully). Rounding a real number α of absolute value at most 1 like this gives us a rational number $\hat{\alpha}$ of the form $a/2^k$ for some integer a with $|a| \leq 2^k$. Moreover, we have that $|\alpha - \hat{\alpha}| < 1/2^k$. We now solve MME on this instance obtained after rounding. The crucial advantage now is that since all the denominators in this rounded instance are 2^k, their LCM is just 2^k, and clearing the denominator no longer incurs a prohibitive increase in the bit complexity. We now invoke the algorithm for exact-MME for integer instances described in the earlier subsection. The details can be found in Section 5.

1.3.4 Algorithm for exact-MME over rationals

For our algorithm for exact-MME, we start by first invoking the algorithm for approximate-MME on the instance for a sufficiently good accuracy parameter t. The choice of t depends upon the output bit complexity that is given to us as a part of the input. From the guarantees on the output of the approximate-MME algorithm, we know that the approximate-MME outputs rational numbers that are at most $1/2^t$ away from the true evaluations. If we can somehow recover the true evaluations from these approximations, we would be done! What we have here are instances of the following problem: our goal is to find a hidden rational number, denoted by a/b (the true evaluation) and we have access to another rational number, denoted by A/B (an approximation to the true evaluation), with the guarantee that $|A/B - a/b| < 1/2^t$ and $|A|, |B| < 2^{O(t)}$. Crucially, we also have a parameter s (given to us as a part of the input) and a guarantee that $|a|, |b| < 2^s$.

This is essentially an instance of rational number reconstruction, which is a well-studied and classical problem of interest in computational algebra and number theory. We rely on these results (essentially in a black-box manner), and in particular the notion and properties of continued fractions to solve this problem efficiently. We observe that our choice of the parameter t (as a function of s) implies that a/b is a *convergent* (a rational number obtained by a truncation of the continued fraction representation of A/B). This observation along with some of the properties of convergents lets us find a/b in nearly-linear time given A/B. The details can be found in Section 6.

2 Preliminaries

Notation

- We will use boldface letters $\mathbf{a}, \mathbf{b}$ etc. for finite-dimensional vectors. We will also use this to denote tuples of variables $\mathbf{x} = (x_1, \ldots, x_n)$ etc. Usually the dimension of the vectors would be clear from context.
- For *exponent vectors* $\mathbf{e} = (e_1, \ldots, e_n) \in \mathbb{Z}^n_{\geq 0}$ and a vector $\mathbf{x} = (x_1, \ldots, x_n)$, we will use $\mathbf{x}^{\mathbf{e}}$ to denote the monomial $x_1^{e_1} \cdots x_n^{e_n}$.
- For a real number α, we use $\lfloor \alpha \rceil$ to denote the closest integer to α. When $\alpha = a + \frac{1}{2}$ for some integer a, $\lfloor \alpha \rceil$ is defined as a.

2.1 Useful inequalities

Lemma 2.1 (Bounds on binomial series)**.** *For $d \in \mathbb{N}$ and $\varepsilon > 0$ with $|\varepsilon| < 1/d^2$, we have*

$$1 + d\varepsilon \leq (1+\varepsilon)^d \leq 1 + d\varepsilon + d^2\varepsilon^2.$$

Proof. The inequalities are clearly true for $d = 1, 2$, so for the rest of this discussion, we assume without loss of generality that $d \geq 3$.

For any $i \geq 3$, we have $\binom{d}{i} \leq \binom{d}{2} \cdot d^{i-2}$. Hence,

$$\left| \sum_{i=3}^{d} \binom{d}{i} \varepsilon^i \right| \leq \sum_{i=3}^{d} \binom{d}{i} \left| \varepsilon^i \right| \leq \binom{d}{2} \cdot \varepsilon^2 \cdot \sum_{i=1}^{d-2} |d\varepsilon|^i < \binom{d}{2} \cdot \varepsilon^2$$

where the last inequality uses $\varepsilon < 1/d^2$. Therefore,

$$(1+\varepsilon)^d = 1 + d\varepsilon + \binom{d}{2}\varepsilon^2 + \sum_{i=3}^{d} \binom{d}{i} \varepsilon^i \geq 1 + d\varepsilon$$

and

$$\begin{aligned}(1+\varepsilon)^d &= 1 + d\varepsilon + \binom{d}{2}\varepsilon^2 + \sum_{i=3}^{d} \binom{d}{i} \varepsilon^i \\ &\leq 1 + d\varepsilon + \binom{d}{2}\varepsilon^2 + \binom{d}{2}\varepsilon^2 \\ &\leq 1 + d\varepsilon + d^2\varepsilon^2.\end{aligned}$$

□

2.2 Kronecker map for base-d

The Kronecker map is a commonly used tool used to perform a variable reduction without changing the underlying sparsity. This map is defined formally as follows.

Definition 2.2 (Kronecker map for base-d)**.** *The c-variate Kronecker map for base-d, denoted by $\Phi_{d,m;c}$ maps cm-variate polynomials into a c-variate polynomials via*

$$\Phi_{d,m;c}(f(x_{1,1}, \ldots, x_{1,m}, \ldots, x_{c,1}, \ldots, x_{c,m})) = f\left(1, y_1^d, y_1^{d^2}, \ldots, y^{d^{m-1}}, \ldots, 1, y_c^d, y_c^{d^2}, \ldots, y_c^{d^{m-1}}\right).$$

If f is a polynomial of individual degree less than d, then the monomial $\mathbf{x}_1^{\mathbf{e}_1} \cdots \mathbf{x}_c^{\mathbf{e}_c}$ is mapped to the monomial $y_1^{e_1} \cdots y_c^{e_c}$ where $\mathbf{e}_i$ is the base-d representation of e_i.

In the same spirit, we define the inverse Kronecker*, denoted by $\Phi^{-1}_{d,m;c}$, that maps a c-variate polynomial of individual degree less than d^m into a cm-variate polynomial of individual degree less than d, given via extending the following map linearly over monomials:*

$$\Phi^{-1}_{d,m;c}(y_1^{e_1} \cdots y_c^{e_c}) = \mathbf{x}_1^{\mathbf{e}_1} \cdots \mathbf{x}_m^{\mathbf{e}_m}$$

where $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,m})$ and $\mathbf{e}_i \in \{0, \ldots, d-1\}^m$ is the base-d representation of $e_i < d^m$.

Associated with the inverse Kronecker map, we also define $\psi_{d,m;c} : \mathbb{F}^c \to \mathbb{F}^{cm}$ that acts on points, given by

$$\psi_{d,m;c} : (a_1, \ldots, a_c) \mapsto (1, a_1^d, \ldots, a_1^{d^{m-1}}, \ldots, 1, a_c^d, \ldots, a_c^{d^{m-1}}).$$

$\Diamond$

The inverse Kronecker map is defined so that we have the following observation.

Observation 2.3 (Kronecker maps and evaluations)**.** *If $f(x_1, \ldots, x_c)$ is a polynomial of individual degree less than d^m, then for any $\mathbf{a} \in \mathbb{F}^c$, we have that $\Phi^{-1}_{d,m;c}(f)(\psi_{d,m;c}(\mathbf{a})) = f(\mathbf{a})$.* □

The above observation would be useful to *trade-off* degree with the number of variables as needed in some of our proofs.

2.3 Computing all primes less than a given number

The classical Prime Number Theorem [Had96, LVP97] asserts that there are $\Theta(N/\log N)$ primes numbers less than N, asymptotically. We can compute all prime numbers less than N in deterministic $\tilde{O}(N)$ time.

Algorithm 1: `PrimeSieve`

Input : An integer $N > 1$

Output: All prime numbers less than N.

```
1 Initialise an array S indexed with 2, 3, ..., N with all values set to TRUE.
2 for i ← 2 to √N do
3     if S[i] is TRUE then
4         Set j ← 2i
5         while j ≤ N do
6             Set S[j] to FALSE.
7             j ← j + i.
8 return {i : S[i] is TRUE}.
```

Lemma 2.4 (Computing primes less than a given number)**.** *There is a deterministic algorithm (Algorithm 1) that computes the set of all primes less N in deterministic time $\tilde{O}(N)$.* □

2.4 Fast Chinese Remaindering

We also rely on the following two theorems concerning fast algorithms for questions related to the Chinese Remainder Theorem (CRT). We refer the reader to the book by von zur Gathen and Gerhard [GG13] for proofs.

Lemma 2.5 (Fast-CRT: moduli computation)**.** *There is an algorithm that, when given as input coprime positive integers $p_1, \ldots, p_r$ and a positive integer N with $N < \prod p_i < 2^c$, computes the remainders $a_i = N \bmod p_i$ for $i = 1, \ldots, r$ in deterministic $\tilde{O}(c)$ time.*

For proof of the above lemma see [GG13, Theorem 10.24].

Lemma 2.6 (Fast-CRT: reconstruction)**.** *There is an algorithm that, when given as input coprime positive integers $p_1, \ldots, p_r$ and $a_1, \ldots, a_r$ such that $0 \leq a_i < p_i$ outputs the unique integer $0 \leq N < \prod p_i$ such that $N = a_i \bmod p_i$ for $i = 1, \ldots, r$ in deterministic $\tilde{O}(c)$ time where $\prod p_i < 2^c$.*

For proof of the above lemma see [GG13, Theorem 10.25]

2.5 Input model for arbitrary precision reals

Throughout this section, we will assume that all real numbers that are "inputs" (namely the coefficients of the polynomial and the coordinates of the evaluation points) are in the range $(-1, 1)$ and are provided via *approximation oracles* with the following guarantees:

Definition 2.7 (Approximation oracle)**.** *The* approximation oracle *for $\alpha \in (-1, 1)$, can provide the "sign" α in $O(1)$ time, and on input k returns an integer $b_k \in [-2^k, 2^k]$ satisfying*

$$\left|\alpha - {}^{b_k}/_{2^k}\right| < {}^1/_{2^k}.$$

We will use $\lfloor \alpha \rceil_k$ to refer to the fraction ${}^{b_k}/_{2^k}$ obtained from the approximation oracle.

The running time of the approximation oracle is the time taken to output b_k. We will say that the approximation oracle is efficient *if the running time is $\tilde{O}(k)$.* ◊

Such efficient approximation oracles can be obtained for any "natural" real number from any sufficiently convergent series. For algebriac reals of the form $\sqrt{2}$ etc., the standard Taylor series is sufficient. Even for "natural" transcendental numbers, we may have such approximation oracles:

$$\begin{aligned}
e &= 1 + \frac{1}{1!} + \frac{1}{2!} + \cdots, \\
\pi &= 4 \cdot \tan^{-1}(1) \\
&= 4 \cdot \left(\tan^{-1}({}^1/_2) + \tan^{-1}({}^1/_3)\right) \\
&= 4 \cdot \left({}^1/_2 - \frac{{}^1/_{2^3}}{3} + \frac{{}^1/_{2^5}}{5} - \cdots \quad + \quad {}^1/_3 - \frac{{}^1/_{3^3}}{3} + \frac{{}^1/_{3^5}}{5} - \cdots\right).
\end{aligned}$$

Any explicit series with $\tilde{O}(k)$ terms of the series having an error less than ${}^1/_{2^k}$ would qualify as an efficient approximation oracle for the purposes of the approximate-MME algorithm over reals.

«SG: PH says 'reorder'. Not sure what he means »

Lemma 2.8 (Repeated exponentiation for approximation oracles)**.** *Given an approximation oracle A for a real number $\alpha \in (-1, 1)$ with running time $T(k)$, and any positive integer D, we can build an approximation oracle A^D for α^D with running time $T(k + O(\log D)) + \tilde{O}(k \log D)$.*

Proof. On an input k, we wish to find an integer $r_k \in [-2^k, 2^k]$ such that $\left|\alpha^D - {}^{r_k}/_{2^k}\right| < {}^1/_{2^k}$.

Let us first consider the case when D is even. Let $t = k + 3$ and suppose we recursively compute an integer $a_t \in [-2^t, 2^t]$ such that $\left|\alpha^{D/2} - {}^{a_t}/_{2^t}\right| < {}^1/_{2^t}$. Let $\delta = {}^{a_t}/_{2^t} - \alpha^{D/2}$.

$$\left|\alpha^D - {}^{a_t^2}/_{2^{2t}}\right| = \left|(\alpha^{D/2})^2 - (\alpha^{D/2} + \delta)^2\right| < 4 \cdot {}^1/_{2^t} \leq {}^1/_{2^{k+1}}$$

Thus, if $R_k = r_k \cdot 2^{2t-k}$ is the multiple of 2^{2t-k} that is closest to a_t^2, then

$$\begin{aligned}\left|\alpha^D - r_k/2^k\right| &\leq \left|\alpha^D - a_t^2/2^{2t}\right| + \left|a_t^2/2^{2t} - r_k 2^{2t-k}/2^{2t}\right| \\ &< 1/2^{k+1} + 2^{2t-k-1}/2^{2t} \leq 1/2^k.\end{aligned}$$

If D is odd, then let $t = k+4$. We use the approximation oracle A to obtain an integer $b_t \in [-2^t, 2^t]$ such that $|\alpha - b_t/2^t| < 1/2^t$, and recursively compute an integer $a_t \in [-2^t, 2^t]$ such that $\left|\alpha^{(D-1)/2} - a_t/2^t\right| < 1/2^t$. Then,

$$\begin{aligned}\left|\alpha^D - a_t^2 b_t/2^{3t}\right| &\leq |\alpha| \left|\left(\alpha^{(D-1)/2}\right)^2 - a_t^2/2^{2t}\right| + \left|a_t^2/2^{2t}\right| |\alpha - b_t/2^t| \\ &< 4 \cdot 1/2^t + 1/2^t \leq 1/2^{k+1}.\end{aligned}$$

Similarly, if $R_t = r_t \cdot 2^{3t-k}$ is the multiple of 2^{3t-k} that is closest to $a_t^2 b_t$, then

$$\left|\alpha^D - r_t/2^k\right| < 1/2^k.$$

If $\mathcal{T}(k, D)$ is the running time of this algorithm (namely Algorithm 2) to compute $r_k \in [-2^k, 2^k]$ such that $\left|\alpha^D - r_k/2^k\right| \leq 1/2^k$, then we have

$$\begin{aligned}\mathcal{T}(k, D) &\leq \mathcal{T}(k+4, D/2) + \tilde{O}(k) \\ &\leq \mathcal{T}(k + O(\log D), 1) + \tilde{O}(k \log D) \\ &= T(k + O(\log D)) + \tilde{O}(k \log D).\end{aligned}$$

□

Algorithm 2: `ApproximationOracle-Powering`

Input : An approximation oracle A for a real number α, an integer $D > 0$, and an integer $k > 0$.

Output: An integer $r_k \in [-2^k, 2^k]$ such that $\left|\alpha^D - r_k/2^k\right| < 1/2^k$.

if $D = 1$ **then**
  return $r_k = A(k)$.
if D *is even* **then**
  Let $t = k + 3$.
  Compute $a_t =$ `ApproximationOracle-Powering`$(A, D/2, t)$.
  return $\lfloor a_t^2/2^{2t-k} \rceil$.
else
  Let $t = k + 4$.
  Compute $b_t = A(t)$. Compute $a_t =$ `ApproximationOracle-Powering`$(A, (D-1)/2, t)$.
  return $\lfloor a_t^2 \cdot b_t/2^{3t-k} \rceil$.

We also note that this notion of approximation oracles naturally extends to representation of

complex numbers. Here, each complex number is given by two such oracles, corresponding to the real and the imaginary part respectively.

3 Revisiting MME over prime fields

We recall the result of Bhargava, Ghosh, Guo, Kumar and Umans [BGGKU22].

Theorem 3.1 (Fast multivariate multipoint evaluation over finite fields [BGGKU22])**.** *There is a deterministic algorithm that when given as input the coefficient vector of an m variate polynomial f of degree less than d in each variable over some finite field $\mathbb{F}$, and N points $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N \in \mathbb{F}^m$ outputs $f(\mathbf{a}_1), f(\mathbf{a}_2), \ldots, f(\mathbf{a}_N)$ in time*

$$(d^m + Nm)^{1+o(1)} \cdot \mathrm{poly}(m, d, \log |\mathbb{F}|),$$

for all $m \in \mathbb{N}$ and sufficiently large $d \in \mathbb{N}$.

The above running time is not *quite* nearly-linear in the input considered as bits due to the factor of $\mathrm{poly}(\log |\mathbb{F}|)$. Also, in the setting when m is a constant, we can no longer absorb $\mathrm{poly}(d)$ within $(Nm + d^m)^{o(1)}$. However, we show below that for the case of prime fields, we can get around these issues and obtain the following nearly linear-time bound.

Theorem 3.2 (Nearly-linear time MME over prime fields)**.** *There is a deterministic algorithm (namely Algorithm 3) that, when given as input the coefficient vector of an m-variate polynomial f of degree less than d in each variable over a prime field $\mathbb{F}_p$, and N points $\mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(N)} \in \mathbb{F}^m$, outputs $f(\mathbf{a}^{(1)}), \ldots, f(\mathbf{a}^{(N)})$ in time*

$$((d^m + Nm) \cdot \log p)^{1+o(1)}$$

for all $m \in \mathbb{N}$ and sufficiently large $d \in \mathbb{N}$.

We first discuss how we handle the two cases when the number of variables is constant and growing with the input respectively in the following two subsections and then prove Theorem 3.2.

We first discuss how we handle the two cases when the number of variables is constant and growing with the input respectively in the following two subsections and then prove Theorem 3.2.

3.1 Handling cases when the number of variables is too small

As mentioned above, in the setting when the number of variables is too small (say $m \leq c$ for a constant c), we may no longer have that $\mathrm{poly}(d) = d^{o(m)}$. However, we can use the inverse-Kronecker map (Definition 2.2) to trade-off degree with the number of variables.

To make the parameters more informative, we rename them and let f be a c-variate polynomial of individual degree less than D, and let $\mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(N)} \in \mathbb{F}_p^c$ be the points at which we wish to evaluate the polynomial.

Let $d = \lfloor \log D \rfloor$ and m be the smallest integer such that $d^m > D$. Note that $d^m > D > d^{m-1}$ and $m = \Theta(\log D / \log\log D)$. If $f(x_1, \ldots, x_c) = \sum_{\mathbf{e}} f_{\mathbf{e}} \cdot \mathbf{x}^{\mathbf{e}}$, define the polynomial $g(y_{1,1}, \ldots, y_{c,m}) = \Phi^{-1}_{d,m;c}(f)$, as defined in Definition 2.2.

For all $i \in [N]$, define $\widetilde{\mathbf{a}^{(i)}} = \psi_{d,m;c}(\mathbf{a}^{(i)})$, as defined in Definition 2.2. Then, from Observation 2.3, we have that $f(\mathbf{a}^{(i)}) = g(\widetilde{\mathbf{a}^{(i)}})$ for all $i \in [N]$. The following observation shows that $\widetilde{\mathbf{a}^{(i)}}$ can be computed efficiently from $\mathbf{a}^{(i)}$.

Observation 3.3. *Given $\mathbf{a} \in \mathbb{F}_p^c$, the point $\widetilde{\mathbf{a}} := \psi^{(c)}_{d,m}(\mathbf{a}) \in \mathbb{F}_p^{cm}$ can be computed in $\mathrm{poly}(d, m, c) \cdot \tilde{O}(\log p)$ time.*

Proof. The running time bound follows from repeated exponentiation as $a^{d^k} \bmod p = (a^{d^{k-1}} \bmod p)^d \bmod p$ and the fact that additions and multiplications modulo p can be performed in $\tilde{O}(\log p)$ time. □

Thus, the task of computing $f(\mathbf{a}^{(1)}), \ldots, f(\mathbf{a}^{(N)})$ reduces to the task of computing the evaluations $g(\widetilde{\mathbf{a}^{(1)}}), \ldots, g(\widetilde{\mathbf{a}^{(N)}})$ where $\widetilde{\mathbf{a}^{(i)}} = \psi^{(c)}_{d,m}(\mathbf{a})$. Also, the reduction runs in time $((D^c + Nc) \cdot \log p)^{1+o(1)}$ since $d, m = D^{o(1)}$.

3.2 When individual degree and number of variables are moderately growing

We return to the familiar variable convention of $f(x_1, \ldots, x_m) \in \mathbb{F}_p[x_1, \ldots, x_m]$ with degree in each variable less than d. From the previous section, may assume without loss of generality that $d, m = \omega(1)$ and hence $\mathrm{poly}(d, m) = (d^m + Nm)^{o(1)}$. Let f be written as a sum of monomials as follows.

$$f(x_1, \ldots, x_m) = \sum_{\mathbf{e}} f_{\mathbf{e}} \cdot x_1^{e_1} \cdots x_m^{e_m}.$$

Interpreting the above as a polynomial over integers with each coefficient in $\{0, 1, \ldots, p-1\}$, and for any $\mathbf{a} \in \{0, \ldots, p-1\}^m$, the integer $f(\mathbf{a})$ is bounded by $d^m \cdot p \cdot p^{dm}$. The idea is to use Chinese Remainder Theorem to reduce the problem to MME over smaller prime fields.

Algorithm 3: `NearlyLinearTimeMME-PrimeFields`

Input : $f(x_1, \ldots, x_m) \in \mathbb{F}_p[x_1, \ldots, x_m]$ with degree in each variable less than d, and $\mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(N)} \in \mathbb{F}_p^m$.

Output: Evaluations $b_i = f(\mathbf{a}^{(i)})$ for $i \in [N]$.

1 **if** $m < \log\log d$ **then**

2 Let $d' = \lfloor \log d \rfloor$ and m' be the smallest integer such that $(d')^{m'} > d$.

3 Replace f by $\Phi^{-1}_{d',m';m}(f)$ and each $\mathbf{a}^{(i)}$ by $\psi_{d',m';m}(\mathbf{a}^{(i)})$.

4 Let $\tilde{L} = (dm+1)\log p + m \log d$. Compute the first $\tilde{L}$ primes numbers $\{p_1, \ldots, p_{\tilde{L}}\}$.

5 Let $L \leq \tilde{L}$ be the smallest integer such that $p_1 \cdots p_L =: M > d^m \cdot p \cdot p^{dm}$.

6 **for** $\mathbf{e} \in \{0, \ldots, d-1\}^m$ **do**

7 Compute $f_{\mathbf{e}}^{(\ell)} = f_{\mathbf{e}} \bmod p_\ell$ for $\ell \in L$ via fast-CRT-moduli-computation (Lemma 2.5).

8 **for** $i \in [N], k \in [m]$ **do**

9 Compute $a_{i,k,\ell} = \mathbf{a}_k^{(i)} \bmod p_\ell$ for $\ell \in L$ via fast-CRT-moduli-computation (Lemma 2.5).

10 **for** $\ell \in L$ **do**

11 Let $f^{(\ell)}(x_1, \ldots, x_m) = \sum_{\mathbf{e}} f_{\mathbf{e}}^{(\ell)} \mathbf{x}^{\mathbf{e}} \in \mathbb{F}_{p_i}[\mathbf{x}]$.

12 Let $\mathbf{a}^{(i,\ell)} = (a_{i,1,\ell}, \ldots, a_{i,m,\ell}) \in \mathbb{F}_{p_\ell}^m$ for each $i \in [N]$.

13 Compute $b_{i,\ell} = f^{(\ell)}(\mathbf{a}^{(i,\ell)})$ for all $i \in [N]$ using Theorem 3.1.

14 **for** $i \in [N]$ **do**

15 Compute the unique $b_i \in [0, M)$ such that $b_i = b_{i,\ell} \bmod p_\ell$ for all $\ell \in [L]$, via fast-CRT-reconstruction (Lemma 2.6).

16 **return** $(b_1 \bmod p, \ldots, b_N \bmod p)$.

Proof of Theorem 3.2. The correctness of Algorithm 3 is evident.

As for the running time, Lines 1 to 3 takes $(d^m + Nm)^{1+o(1)}$ time by Observation 3.3 and reduces to the case when $m \geq \log\log d$. In this case, Lines 4 and 5 require $\tilde{O}(\tilde{L})$ time (Lemma 2.4), which is $\tilde{O}(\log p) \cdot \mathrm{poly}(d, m)$.

Using Lemma 2.5, we have that Lines 6 to 9 require time $(d^m + Nm) \cdot \tilde{O}(\log M) = ((d^m + Nm) \cdot \log p)^{1+o(1)}$.

From Theorem 3.1, we have that Line 13 runs in time $(d^m + Nm)^{1+o(1)} \cdot \mathrm{poly}(d, m, \log p_i)$, and since $p_i < \tilde{O}(\tilde{L}) = \tilde{O}(dm \log p)$, the entire loop in Lines 10 to 13 takes time $(d^m + Nm)^{1+o(1)} \cdot \tilde{O}(\log p) = ((d^m + Nm)\log p)^{1+o(1)}$.

And finally, from Lemma 2.6 we have that the entire loop in Lines 14 to 15 takes time $(Nm) \cdot \tilde{O}(\log M) = ((d^m + Nm) \cdot \log p)^{1+o(1)}$. Hence, Algorithm 3 runs in time $((d^m + Nm)\log p)^{1+o(1)}$. □

4 Exact-MME over integers with known output bit complexity

In this section, we study the following version of MME over integers.

Input: An integer $s > 0$, a polynomial $f(x_1, \ldots, x_m) \in \mathbb{Z}[x_1, \ldots, x_m]$ of individual degree less than d, given as a list of d^m integer coefficients, a set of points $\mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(N)} \in \mathbb{Z}^m$ with each coordinate of magnitude at most 2^s, with the guarantee that all coefficients of f, coordinates of $\mathbf{a}^{(i)}$'s, and evaluations $f(\mathbf{a}^{(i)})$ are bounded in magnitude by 2^s.

Output: Integers $b_1, \ldots, b_N$ that are the evaluations, i.e. $b_i = f(\mathbf{a}^{(i)})$ for $i \in [N]$.

Theorem 4.1 (Exact-MME over integers)**.** *There is a deterministic algorithm (namely Algorithm 4) that on input as mentioned above returns the required output as mentioned above and runs in deterministic time* $((d^m + Nm) \cdot s)^{1+o(1)}$ *for all* $m \in \mathbb{N}$ *and sufficiently large* $d \in \mathbb{N}$.

The main idea is to use the Chinese Remainder Theorem and reduce to the case of MME over finite fields. Since we wish to obtain a nearly-linear time algorithm, we would once again need to use Chinese Remainder Theorem implemented in nearly-linear time (Lemmas 2.5 and 2.6) and make use of the nearly-linear time algorithm for MME over prime fields (Theorem 3.2).

Algorithm 4: `ExactMME-integers`

Input : $f(x_1, \ldots, x_m) \in \mathbb{Z}[x_1, \ldots, x_m]$ and $\mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(N)} \in \mathbb{Z}^n$, and an integer $s > 0$ such that all coefficients of f, coordinates of $\mathbf{a}^{(i)}$ and evaluations $f(\mathbf{a}^{(i)})$ have magnitude bounded by 2^s.
Output: Evaluations $b_i = f(\mathbf{a}^{(i)})$ for $i \in [N]$.

Compute the first s primes numbers $\{p_1, \ldots, p_s\}$.
Let $L \leq s$ be the smallest integer such that $p_1 \cdots p_L =: M > 2^{s+1}$.
for $\mathbf{e} \in \{0, \ldots, d-1\}^m$ **do**
    Compute $f_{\mathbf{e}}^{(\ell)} = f_{\mathbf{e}} \bmod p_\ell$ for $\ell \in L$ using Lemma 2.5.
for $i \in [N], k \in [m]$ **do**
    Compute $a_{i,k,\ell} = \mathbf{a}_k^{(i)} \bmod p_\ell$ for $\ell \in L$ using Lemma 2.5.
for $\ell \in [L]$ **do**
    Let $f^{(\ell)}(x_1, \ldots, x_m) = \sum_{\mathbf{e}} f_{\mathbf{e}}^{(\ell)} \mathbf{x}^{\mathbf{e}} \in \mathbb{F}_{p_i}[\mathbf{x}]$.
    Let $\mathbf{a}^{(i,\ell)} = (a_{i,1,\ell}, \ldots, a_{i,m,\ell}) \in \mathbb{F}_{p_\ell}^m$ for each $i \in [N]$.
    Compute $b_{i,\ell} = f^{(\ell)}(\mathbf{a}^{(i,\ell)})$ for all $i \in [N]$ using Algorithm 3.
for $i \in [N]$ **do**
    Compute the unique $b_i \in [-M/2, M/2]$ such that $b_i = b_{i,\ell} \bmod p_\ell$ for all $\ell \in [L]$, using Lemma 2.6.
return $\{b_i : i \in [N]\}$.

Proof of Theorem 4.1. We are guaranteed that $\left|f(\mathbf{a}^{(i)})\right| < 2^s$ for all $i \in [N]$. Hence, by the Chinese Remainder Theorem, it is sufficient to compute $f(\mathbf{a}) \bmod p_i$ for each $i \in [L]$ since $p_1 \cdots p_L > 2^{s+1}$. Hence, the correctness of Algorithm 4 is evident. As for the running time, we will do an analysis very similar to the analysis for Algorithm 3.

Using Lemma 2.5, we have that Lines 1 to 2 require time $O(\tilde{s})$. By the Prime Number Theorem [Had96, LVP97], we also have that each $p_i = \tilde{O}(s)$ and hence $p_1 \cdots p_L < 2^{s+1} \cdot \tilde{O}(s)$.

From Theorem 3.1, we have that Line 10 runs in time $((d^m + Nm) \cdot \log p_\ell)^{1+o(1)}$ the entire loop in Lines 7 to 10 takes time $((d^m + Nm)(\sum_\ell \log p_\ell))^{1+o(1)} = ((d^m + Nm) \cdot s)^{1+o(1)}$.

And finally, from Lemma 2.6 we have that the entire loop in Lines 11 to 12 takes time $(Nm) \cdot \tilde{O}(\log M) = ((d^m + Nm) \cdot s)^{1+o(1)}$. Hence, Algorithm 4 runs in time $((d^m + Nm) \cdot s)^{1+o(1)}$ as claimed. □

Remark 4.2. If we are only given that all coefficients of f and all coordinates of the points are integers bounded in magnitude by 2^s with no a-priori bound on the bit complexity of the evaluations, a naïve bound on the size of evaluations is

$$|f(\mathbf{a})| \leq d^m \cdot 2^s \cdot 2^{sdm} \leq 2^{sdm+s+m\log d}.$$

Thus, we may use $s' = (sdm + s + m \log d)$ in Theorem 4.1 to get the time complexity bounded by $((d^m + Nm) \cdot (sdm))^{1+o(1)}$. If m is a growing function, then the output complexity is nearly-linear in the input complexity since $\text{poly}(d) = (d^m + Nm)^{o(1)}$. But, in the regime when m is a constant, this is super-linear in the input size $(d^m + Nm) \cdot s$ because of the additional factor of d. However, a slightly worse running time is to be expected in this case since the output complexity is $\Omega(N \cdot sdm)$ in the worst case. ♢

5 Approximate-MME over reals

Throughout this section, we will assume that all real numbers as part of the input are in the interval $(-1, 1)$.

Remark 5.1 (On the restriction on absolute value of constants)**.** *Given any arbitrary polynomial $f(\mathbf{x}) \in \mathbb{R}[\mathbf{x}]$, we can scale the polynomial by the largest coefficient to obtain and run the approximate-MME on the scaled polynomial $\tilde{f}$. If we have $|\tilde{f}(\mathbf{a}) - \beta_i| \leq \varepsilon$, then we immediately have $|f(\mathbf{a}) - (\max |f_\mathbf{e}|)\, \beta_i| \leq \varepsilon \cdot (\max |f_\mathbf{e}|)$. Thus, we may assume without loss of generality that all coefficients of f have absolute value at most* 1.

However, the assumption that coordinates of all evaluation points have absolute value bounded by one is not *without loss of generality but is well-motivated nevertheless. Even in the case of univariate integer polynomials, the evaluation $f(\mathbf{a})$ could be as large as $|\mathbf{a}|^d$ where $|\mathbf{a}| = \max |\mathbf{a}_i|$. Therefore, the output bit-complexity for MME is potentially $O(d \cdot N)$ which is super-linear in the input bit-complexity.*

The restriction of insisting that evaluation points consist of coordinates with absolute value at most 1 *ensures that the evaluations are never prohibitively large in magnitude, thereby making the quest for approximate-MME in nearly-linear time more meaningful.* ♢

5.1 The problem statement and algorithm

We now state the precise problem statement and our results for approximate-MME over the field of real numbers.

> **Input:** A polynomial $f(x_1, \ldots, x_m) \in \mathbb{R}_{(-1,1)}[x_1, \ldots, x_m]$ of individual degree less than d, given as a list of d^m efficient approximation oracles for each coefficient, a set of points $\mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(N)} \in (-1,1)^m$ each of whose coordinates are also provided via efficient approximation oracles, and an accuracy parameter t.
>
> **Output:** Rational numbers $b_1, \ldots, b_N$ such that $\left| f(\mathbf{a}^{(i)}) - b_i \right| < 1/2^t$ for all $i \in [N]$.

Theorem 5.2 (approximate-MME over reals)**.** *There is a deterministic algorithm (namely Algorithm 5) that on input as mentioned above returns the required output as mentioned above and runs in time* $((d^m + Nm) \cdot t)^{1+o(1)}$ *for all* $m \in \mathbb{N}$ *and sufficiently large* $d \in \mathbb{N}$.

The rest of the section is devoted to the proof of the above theorem.

High-level idea: The algorithm is a suitable reduction to the task of exact-MME over integers (Theorem 4.1). We will replace each of the real numbers by appropriately chosen approximations of the form $a_i/2^k$ (for a suitable large $k = O(t)$) so that the evaluations of the perturbed polynomial at the perturbed points are not too far from the original evaluations. Since we now have all denominators of the form 2^k, we can *clear* the denominators and reduce to the case of computing MME over integers.

As expected, there are some subtleties that need to be addressed to make sure that the entire algorithm runs in nearly-linear time.

Rounding coefficients of f

Let k be a parameter to be chosen shortly. Define the polynomial $\lfloor f \rceil_k$ as

$$\lfloor f \rceil_k(x_1, \ldots, x_m) := \sum_{\mathbf{e}} \lfloor f_{\mathbf{e}} \rceil_k \cdot \mathbf{x}^{\mathbf{e}}.$$

Observation 5.3 (Error due to rounding coefficients of f)**.** *For any* $\mathbf{a} \in (-1,1)^m$*, we have that*

$$|f(\mathbf{a}) - \lfloor f \rceil_k(\mathbf{a})| \leq 1/2^{k - m \log d}.$$

Proof.

$$\begin{aligned} & f(\mathbf{a}) - \lfloor f \rceil_k(\mathbf{a}) = \sum_{\mathbf{e}} (f_{\mathbf{e}} - \lfloor f_{\mathbf{e}} \rceil_k) \cdot \mathbf{a}^{\mathbf{e}} \\ \implies & |f(\mathbf{a}) - \lfloor f \rceil_k(\mathbf{a})| \leq \sum_{\mathbf{e}} |f_{\mathbf{e}} - \lfloor f_{\mathbf{e}} \rceil_k| \cdot |\mathbf{a}^{\mathbf{e}}| \leq d^m \cdot 1/2^k. \end{aligned}$$

□

Rounding points

Let k be a parameter to be chosen shortly. For any $\mathbf{a} = (a_1, \ldots, a_m) \in (-1,1)^m$, define $\lfloor \mathbf{a} \rceil_k$ as

$$\lfloor \mathbf{a} \rceil_k := (\lfloor a_1 \rceil_k, \ldots, \lfloor a_m \rceil_k) .$$

Observation 5.4 (Error due to rounding points)**.** *Let* $\mathbf{e} = (e_1, \ldots, e_m) \in \{0, \ldots, d-1\}^m$ *and* $\mathbf{a} \in (-1,1)^m$*. Suppose* $k \in \mathbb{N}$ *such that* $2^k > 4d^2m^2$*. Then,*

$$\left| \mathbf{a}^{\mathbf{e}} - \lfloor \mathbf{a} \rceil_k^{\mathbf{e}} \right| \leq 1/2^{k - \log(4dm)}$$

Proof. Note that all $a_i \in (-1,1)$. Let $\delta_i = \lfloor a_i \rceil_k - a_i$ for $i \in [m]$; we have that $|\delta_i| \leq 1/2^k \leq 1/4d^2m^2$. Hence,

$$\begin{aligned} \lfloor a_1 \rceil_k^{e_1} \cdots \lfloor a_m \rceil_k^{e_m} &= (a_1 + \delta_1)^{e_1} \cdots (a_m + \delta_m)^{e_m} \\ &= a_1^{e_1} \cdots a_m^{e_m} + \sum_{\substack{j_1 \leq e_1, \ldots, j_m \leq e_m \\ \text{not all } j_i = 0}} \binom{e_1}{j_1} \cdots \binom{e_m}{j_m} \cdot \prod_{i=1}^{m} \left(a_i^{e_i - j_i} \cdot \delta_i^{j_i} \right) \end{aligned}$$

$$\begin{aligned} \implies \left| \lfloor a_1 \rceil_k^{e_1} \cdots \lfloor a_m \rceil_k^{e_m} - a_1^{e_1} \cdots a_m^{e_m} \right| &\leq \left| \sum_{\substack{j_1 \leq e_1, \ldots, j_m \leq e_m \\ \text{not all } j_i = 0}} \binom{e_1}{j_1} \cdots \binom{e_m}{j_m} \cdot \prod_{i=1}^{m} \delta_i^{j_i} \right| \\ &\leq \left| \prod_{i=1}^{m} (1 + \delta_{j_i})^d - 1 \right| \\ &\leq (1 + 2d(1/2^k))^m - 1 \leq 4dm(1/2^k). \quad \text{(Lemma 2.1)} \end{aligned}$$

□

Handling the case when number of variables is too small

To make the variables suggestive, we will rename them and say $f(x_1, \ldots, x_c)$ is a c-variate polynomial in $\mathbb{R}_{(-1,1)}[x_1, \ldots, x_c]$ with degree in each variable less than D. We wish to evaluate the polynomial on points $\mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(N)} \in (-1,1)^c$.

Once again, let $d = \lfloor \log D \rfloor$ and let m be the smallest integer such that $d^m > D$. Note that $d^m > D \geq d^{m-1}$ and $m = \Theta(\log D / \log\log D)$. Define the polynomial $g(y_{1,1}, \ldots, y_{c,m}) = \Phi_{d,m;c}^{-1}(f)$, as defined in Definition 2.2. Define $\widetilde{\mathbf{a}^{(i)}} = \psi_{d,m;c}(\mathbf{a}^{(i)})$. From Observation 2.3, we have that $f(\mathbf{a}^{(i)}) = g(\widetilde{\mathbf{a}^{(i)}})$ for all $i \in [N]$.

Even if $\mathbf{a}^{(i)}$ consisted of only rational numbers, unlike the setting in Theorem 3.2 where we could use Observation 3.3, the rational numbers in $\widetilde{\mathbf{a}^{(i)}}$ have much larger bit complexity due to the exponentiation. However, by Lemma 2.8, we have efficient approximation oracles for $\widetilde{\mathbf{a}^{(i)}}$ and that suffices for our algorithm.

5.2 Reduction to exact-MME over integers

From the previous subsection, we may now assume without loss of generality that we are working with an m-variate polynomial $f(x_1, \ldots, x_n)$ of individual degree less than d, with both m, d as growing parameters, and wish to evaluate this polynomial on N points $\mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(N)} \in (-1,1)^m$, with all coefficients and coordinates provided via approximation oracles running in time $\tilde{O}(k + O(m \log d))$. We wish to compute integers $b_1, \ldots, b_N$ such that $\left| f(\mathbf{a}^{(i)}) - b_i/2^t \right| < 1/2^t$. We now describe the algorithm (Algorithm 5).

Algorithm 5: `approximate-MME-Reals`

Input : An m-variate polynomial $f(x_1, \ldots, x_m) \in \mathbb{R}_{(-1,1)}[\mathbf{x}]$ of individual degree less than d, and points $\mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(N)} \in \mathbb{R}^m_{(-1,1)}$ (with all real numbers provided via approximation oracles) and an integer $t > 0$.

Output: Integers $b_1, \ldots, b_N$ such that $\left| f(\mathbf{a}^{(i)}) - b_i/2^t \right| < 1/2^t$ for all $i \in [N]$.

1 **if** $m < \log\log d$ **then**

2 Let $d' = \lfloor \log d \rfloor$ and m' be the smallest integer such that $(d')^{m'} > d$.

3 Replace f by $\Phi^{-1}_{d',m';m}(f)$ and each $\mathbf{a}^{(i)}$ by $\psi_{d',m';m}(\mathbf{a}^{(i)})$.

4 Let $k_1 = \lceil t + m \log d + 2 \rceil$ and $k_2 = \lceil t + m \log d + \log(4md) + 2 \rceil$; let $k = \max(k_1, k_2) = k_2$.

5 Compute $\lfloor f \rceil_{k_1} = \sum_{\mathbf{e}} g_{\mathbf{e},k_1}/2^{k_1} \cdot \mathbf{x}^{\mathbf{e}} = 1/2^{k_1} \cdot \sum_{\mathbf{e}} g_{\mathbf{e},k_1} \cdot \mathbf{x}^{\mathbf{e}}$.

6 **for** $i \in [N]$ **do**

7 Compute $\lfloor \mathbf{a}^{(i)} \rceil_{k_2} = (a_{i,1,k_2}/2^{k_2}, \ldots, a_{i,m,k_2}/2^{k_2}) = 1/2^{k_2} \cdot (a_{i,1,k_2}, \ldots, a_{i,m,k_2})$.

8 Let $\widehat{\mathbf{a}^{(i)}} = (a_{i,1,k_2}, \ldots, a_{i,m,k_2})$.

9 Compute the polynomial $G(x_1, \ldots, x_m)$ defined as

$$G(x_1, \ldots, x_n) = \sum_{\mathbf{e} \in \{0,\ldots,d-1\}^m} g_{\mathbf{e},k_1} \cdot 2^{(k_2 dm) - k_2|\mathbf{e}|} \cdot \mathbf{x}^{\mathbf{e}}$$

where $|\mathbf{e}|$ refers to the sum of the coordinates (i.e., the degree of the monomial $\mathbf{x}^{\mathbf{e}}$).

10 Run Algorithm 4 (`Exact-MME-integers`) with inputs $\left(G, \left(\widehat{\mathbf{a}^{(1)}}, \ldots, \widehat{\mathbf{a}^{(N)}}\right), s = 3kdm\right)$ to obtain $B_1, \ldots, B_N$ such that, for all $i \in [N]$, we have

$$B_i = G(\widehat{\mathbf{a}^{(i)}}).$$

Let $b_i = \lfloor B_i/2^{k_1 + k_2 dm - t} \rceil$ for each $i \in [N]$.

11 **return** $(b_1, \ldots, b_N)$.

Proof of correctness: Without loss of generality, we may assume that d, m are growing parameters (from Lines 1 to 3).

Note that for any $\mathbf{a}^{(i)}$, we have

$$\left| f(\mathbf{a}^{(i)}) - \lfloor f \rceil_{k_1}(\lfloor \mathbf{a}^{(i)} \rceil_{k_2}) \right| \leq \left| f(\mathbf{a}^{(i)}) - \lfloor f \rceil_{k_1}(\mathbf{a}^{(i)}) \right| + \left| \lfloor f \rceil_{k_1}(\mathbf{a}^{(i)}) - \lfloor f \rceil_{k_1}(\lfloor \mathbf{a}^{(i)} \rceil_{k_2}) \right|$$

$\leq 1/2^{t+2} + 1/2^{t+2} \leq 1/2^{t+1}$.

«SG: Not sure why cref is unable to combine » where the last inequality uses Observation 5.3 and Observation 5.4 with our choice of k_1 and k_2. Thus, it suffices to compute $\lfloor f \rceil_{k_1}\left(\lfloor \mathbf{a}^{(i)} \rceil_{k_2}\right)$ for each $i \in [N]$. The polynomial $\lfloor f \rceil_{k_1}$ is computed in Line 5 and the points $\lfloor \mathbf{a}^{(i)} \rceil_{k_2}$ are computed in Lines 6 to 8. Let $\widehat{\mathbf{a}^{(i)}} = 2^{k_2} \lfloor \mathbf{a}^{(i)} \rceil_{k_2} \in (-2^{k_2}, 2^{k_2})^m$.
Since each coefficient of $2^{k_1} \cdot \lfloor f \rceil_{k_1}$ is bounded in magnitude by 2^{k_1}, we have

$$\left|G(\widehat{\mathbf{a}^{(i)}})\right| = \left|\sum_{\mathbf{e}\in\{0,\ldots,d-1\}^m} g_{\mathbf{e},k_1} \cdot 2^{(k_2 dm)-k_2|\mathbf{e}|} \cdot \widehat{\mathbf{a}^{(i)}}^{\mathbf{e}}\right| \leq d^m \cdot 2^{k_1} \cdot 2^{k_2 dm} \cdot 2^{k_2 dm} \leq 2^{3kdm}.$$

From the definition of $G(x_1, \ldots, x_m)$, note that

$$\begin{aligned} G(\widehat{\mathbf{a}^{(i)}}) &= \sum_{\mathbf{e}\in\{0,\ldots,d-1\}^m} g_{\mathbf{e},k_1} \cdot 2^{(k_2 dm)-k_2|\mathbf{e}|} \cdot \widehat{\mathbf{a}^{(i)}}^{\mathbf{e}} \\ &= \sum_{\mathbf{e}\in\{0,\ldots,d-1\}^m} g_{\mathbf{e},k_1} \cdot 2^{(k_2 dm)} \cdot \left(1/2^{k_2} \cdot \widehat{\mathbf{a}^{(i)}}\right)^{\mathbf{e}} \\ &= 2^{(k_2 \cdot d \cdot m)} \cdot \sum_{\mathbf{e}\in\{0,\ldots,d-1\}^m} g_{\mathbf{e},k_1} \cdot \lfloor \mathbf{a}^{(i)} \rceil_{k_2}^{\mathbf{e}} \\ &= 2^{k_1+k_2 dm} \cdot \lfloor f \rceil_{k_1}(\lfloor \mathbf{a}^{(i)} \rceil_{k_2}). \end{aligned}$$

Since Theorem 4.1 correctly computes the evaluations of $G(\mathbf{x})$ on $\widehat{\mathbf{a}^{(i)}}$'s, we have we have for each $i \in [N]$

$$1/2^{k_1+k_2 dm} \cdot G\left(\widehat{\mathbf{a}^{(i)}}\right) = \lfloor f \rceil_{k_1}(\lfloor \mathbf{a}^{(i)} \rceil_{k_2}) = B_i/2^{k_1+k_2 dm}.$$

Finally, if $b_i = \lfloor B_i/2^{k_1+k_2 dm-t} \rceil$, then

$$\left|b_i/2^t - B_i/2^{k_1+k_2 dm}\right| = 1/2^t \cdot \left|b_i - B_i/2^{k_1+k_2 dm-t}\right| \leq 1/2^{t+1}.$$

Hence,

$$\left|f(\mathbf{a}^{(i)}) - b_i/2^t\right| \leq \left|f(\mathbf{a}^{(i)}) - \lfloor f \rceil_{k_1}(\lfloor \mathbf{a}^{(i)} \rceil_{k_2})\right| + \left|\lfloor f \rceil_{k_1}(\lfloor \mathbf{a}^{(i)} \rceil_{k_2}) - b_i/2^t\right| \leq 1/2^t.$$

Running time analysis: After Lines 1 to 3, we may assume that $d, m = \omega(1)$ and all coefficients of f and coordinates of points are provided via approximation oracles with running time $\tilde{O}(r + m \log d)$ to compute an r-bit approximation.

Lines 5 to 8 overall takes time

$$(d^m + Nm) \cdot \tilde{O}(k + O(m \log d)) = (d^m + Nm) \cdot \tilde{O}(t + O(m \log d)) = ((d^m + Nm) \cdot t)^{1+o(1)}.$$

Computing the coefficients of $G(\mathbf{x})$ takes time $(d^m) \cdot \tilde{O}(kdm)$. By Theorem 4.1, Line 10 takes time

$$((d^m + Nm) \cdot 3kdm)^{1+o(1)} = ((d^m + Nm) \cdot t)^{1+o(1)}.$$

Therefore, Algorithm 5 takes $((d^m + Nm) \cdot t)^{1+o(1)}$ overall.

This completes the proof of Theorem 5.2. □

6 Exact-MME over rationals with known output complexity

We now use our algorithm for approximate-MME over real numbers to obtain a fast algorithm for exact-MME over the field of rational numbers. We start by formally stating the precise problem that we solve and then build upon some necessary preliminaries that we need for our algorithm.

6.1 The problem statement

Input: A polynomial $f(x_1, \ldots, x_m) \in \mathbb{Q}_{(-1,1)}[x_1, \ldots, x_m]$ of individual degree less than d, given as a list of d^m, a list of points $\mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(N)} \in \mathbb{Q}^m_{(-1,1)}$, an integer parameter $s > 0$ such that all rational numbers in the coefficients of f, the coordinates of points and evaluations $f(\mathbf{a}^{(i)})$ are expressible as rational numbers of the form p/q with $|p|, |q| < 2^s$.

Output: Integers $b_1, \ldots, b_N, c_1, \ldots, c_N$ such that $f(\mathbf{a}^{(i)}) = b_i/c_i$ for all $i \in [N]$.

Theorem 6.1 (Exact-MME over rationals)**.** *There is a deterministic algorithm (namely Algorithm 7) that on input as mentioned above returns the required output as mentioned above and runs in time* $((d^m + Nm) \cdot s)^{1+o(1)}$ *for all* $m \in \mathbb{N}$ *and sufficiently large* $d \in \mathbb{N}$.

Main idea: The main idea would be a reduction to approximate-MME (Theorem 5.2) followed by a *rational number reconstruction* step. If we can compute $f(\mathbf{a}^{(i)})$ to a reasonable degree of accuracy (depending on the output guarantee s), we can recover the rational number exactly from it. Before we present the algorithm for the above theorem, we discuss the notion of continued fractions which would be the key to reconstructing the rational number of interest.

6.2 Continued fractions, rational approximations, and extended Euclid's algorithm

Definition 6.2 (Continued fractions)**.** *A* finite continued fraction *expressed by a sequence of integers* $[q_1, \ldots, q_t]$ *computes the rational number*

$$q_1 + \cfrac{1}{q_2 + \cfrac{1}{\ddots + \cfrac{1}{q_{t-1} + \cfrac{1}{q_t}}}}.$$

An infinite continued fraction *expressed by an infinite sequence of integers* $[q_1, q_2, \ldots]$ *satisfying*[4] $q_2, \ldots, q_n > 0$ *is said to compute a real number* α *if*

$$\alpha = q_1 + \cfrac{1}{q_2 + \cfrac{1}{q_3 + \cfrac{1}{\ddots}}}.$$

in the sense that $\lim_{n\to\infty} |\alpha - [q_1, \ldots, q_n]| = 0$. ◊

We note some basic properties of continued fractions which may be found in most standard texts (cf. Schmidt [Sch80, Chapter 1]).

Proposition 6.3 (Uniqueness of continued fractions (Lemma 4C, 4D in [Sch80]))**.** *Every real number has a unique continued fraction expansion up to the following exceptions:*

1. *If* α *is an integer, then there are exactly two continued fraction representations for* α *namely* $[\alpha]$ *and* $[\alpha - 1, 1]$.
2. *If* α *is a non-integral rational number, then there are exactly two continued fraction representations for* α*: one of the form* $[q_1, \ldots, q_n]$ *with* $q_n \geq 2$*, and* $[q_1, \ldots, q_n - 1, 1]$ *being the other.*
3. *If* α *is irrational, then there is exactly one continued fraction representation for* α.

Definition 6.4 (Convergents)**.** *For a real number* α *with* $[q_1, q_2, \ldots]$ *being the unique*[5]*, the rational number* a_i/b_i *corresponding to the* i*-th prefix* $[q_1, \ldots, q_i]$ *is called* the i-th convergent of α. ◊

Lemma 6.5 (Properties of convergents)**.** *Suppose* $\{a_i/b_i\}_i$ *be the convergents of a real number* $\alpha = [q_1, q_2, \ldots]$*. Then*

1. *For any* $n \geq 3$*, we have*

$$a_n = q_n a_{n-1} + a_{n-2},$$
$$b_n = q_n b_{n-1} + b_{n-2}.$$

   In particular, the denominator sequence $\{b_n\}_{n\geq 2}$ *is increasing.*

2. *For all* $n \geq 1$,

$$\frac{a_{n+1}}{b_{n+1}} - \frac{a_n}{b_n} = \frac{(-1)^{n-1}}{b_n(q_{n+1}b_n + b_{n-1})} = \frac{(-1)^{n-1}}{b_n b_{n+1}}.$$

3. *For any* $n \geq 1$*, unless* $\alpha = \frac{a_n}{b_n}$*, we have*

$$\frac{1}{b_n(b_n + b_{n+1})} \leq \left|\alpha - \frac{a_n}{b_n}\right| \leq \frac{1}{b_n b_{n+1}}.$$

[4]Traditionally, continued fractions with this condition are called 'simple' continued fractions but we will drop this qualifier as we will only deal with continued fractions with this additional constraint.

[5]As a convention, for rational numbers, we will only consider continued fraction representations of the first form described in Proposition 6.3 Items 1 and 2.

4. *Suppose a/b is a rational number satisfying $|\alpha - a/b| < 1/2b^2$ Then, a/b is one of the convergents of α.*

Proof. Items 1, 2 and 4 are just [Sch80, Lemma 3A, Lemma 3E, Theorem 5C] respectively.

For Item 3, if $\alpha \neq a_n/b_n$, we have that q_{n+1} exists. Let $\alpha_{n+1} = [q_{n+1}, \ldots]$. Then, we may abuse notation and express α as the "continued fraction" $\alpha = [q_1, \ldots, q_n, \alpha_{n+1}]$. Item 2 for this expression yields

$$\left|\alpha - \frac{a_n}{b_n}\right| = \frac{1}{b_n(\alpha_{n+1}b_n + b_{n-1})}.$$

Note that $q_{n+1} \leq \alpha_{n+1} \leq q_{n+1} + 1$ and hence

$$\begin{aligned} \left|\alpha - \frac{a_n}{b_n}\right| &= \frac{1}{b_n(\alpha_{n+1}b_n + b_{n-1})} \\ &\leq \frac{1}{b_n(q_{n+1}b_n + b_{n-1})} = \frac{1}{b_n b_{n+1}}, \quad \text{(by Item 1)} \\ \text{and} \quad \left|\alpha - \frac{a_n}{b_n}\right| &= \frac{1}{b_n(\alpha_{n+1}b_n + b_{n-1})} \\ &\geq \frac{1}{b_n(q_{n+1}b_n + b_{n-1} + b_n)} = \frac{1}{b_n(b_n + b_{n+1})}. \end{aligned}$$

□

Extended Euclid's Algorithm

Closely related to continued fractions is the classical Extended Euclid's Algorithm for computing the greatest common divisor of two numbers.

Definition 6.6 (Remainder and quotient sequences)**.** *For a pair of integers $a, b > 0$, we define the* remainder sequence $\{r_i\}_{i=0,\ldots,t+1}$ *and the* quotient sequence $\{q_i\}_{i=1,\ldots,t}$ *for the pair (a, b) as follows:*

- *$r_0 = a$ and $r_1 = b$,*
- *For all $i \geq 1$, define q_i, r_{i+1} as the quotient and remainder respectively when r_{i-1} is divided by r_i. Thus,*

$$r_{i+1} = r_{i-1} \bmod r_i = r_{i-1} - q_i r_i.$$

- *r_{t+1} is the first element of the sequence that is equal to zero.* ◊

Observation 6.7 (Continued fractions for a rational number and quotient sequences)**.** *Suppose $a, b > 0$ are a pair of integers and $\{q_1, \ldots, q_t\}$ is the associated quotient sequence. Then, the continued fraction representation of the rational number a/b is $[q_1, \ldots, q_t]$:*

$$\frac{a}{b} = q_1 + \cfrac{1}{q_2 + \cfrac{1}{\ddots + \frac{1}{q_t}}}.$$

□

Computing the gcd of two given integers, and more generally computing the entire quotient sequence, can be done in deterministic nearly-linear time; this is attributed to Knuth and Schönhage (cf. Möller [Möl08] for a complete description and a detailed history).

Theorem 6.8 (Fast Extended Euclid Algorithm (cf. Möller [Möl08]))**.** *There is a deterministic algorithm that, on input a pair of integers $a > b > 0$ with $a, b \leq 2^s$, computes the entire quotient sequence $q_1, \ldots, q_t$ for the pair (a, b) in time $\tilde{O}(s)$.*

Corollary 6.9 (Fast computation of convergents)**.** *There is a deterministic algorithm that, on input a pair of integers $M, N > 0$ with $M, N \leq 2^s$, and an integer $i > 0$ computes integers a_i, b_i such that a_i/b_i is the i-th convergent of the rational number M/N, with running time $\tilde{O}(s)$.*

Proof. Let $q_1, \ldots, q_t$ be the quotient sequence for the pair (M, N), which may be computed using Theorem 6.8 in $\tilde{O}(s)$ time. By Observation 6.7, this is the continued fraction representation of M/N. Thus, it is easy to note that

$$\begin{bmatrix} a_i & a_{i-1} \\ b_i & b_{i-1} \end{bmatrix} = \begin{bmatrix} q_1 & 1 \\ 1 & 0 \end{bmatrix} \cdots \begin{bmatrix} q_i & 1 \\ 1 & 0 \end{bmatrix}$$

where a_j/b_j is the j-th convergent. Note that $|q_j| < r_{j-1}/r_j$ where $\{r_0, \ldots, r_t\}$ is the associated remainder sequence and hence we have $|q_1 \cdots q_t| \leq M \leq 2^s$. Thus, this matrix product can be computed in $\tilde{O}(s)$ time. □

6.3 Rational number reconstruction

Lemma 6.10 (Fast rational number reconstruction)**.** *There is a deterministic algorithm (namely Algorithm 6) that, given as input an integer parameter $s > 0$ and integers A, B with the guarantee that $|B| < 2^{2s+1}$ and there exist a unique rational number (in reduced form) a/b with $|b| < 2^s$ and*

$$\left| \frac{A}{B} - \frac{a}{b} \right| < \frac{1}{2^{2s+1}},$$

finds the integers a, b in time $\tilde{O}(s)$.

Proof. The algorithm is straightforward given Corollary 6.9 and Lemma 6.5.

Algorithm 6: `Fast-Rational-Number-Reconstruction`

Input : Integers A, B and an integer parameter $s > 0$ such that $|A|, |B| \leq 2^{2s+1}$ and there is some rational number a/b such that $|b| < 2^s$ and $|A/B - a/b| < 1/2^{2s+1}$.

Output: The integers a, b.

1 Using Theorem 6.8, compute the quotient sequence $q_1, \ldots, q_\ell$ for the pair A, B.

2 Using Corollary 6.9 and binary search, compute the largest index i such that the i-th convergent a_i/b_i satisfies $|b_i| < 2^s$.

3 **return** a_i, b_i.

The running time of the algorithm is clearly $\tilde{O}(s)$ as claimed as $\ell = O(\log(A+B)) = O(s)$ and thus we have at most $O(\log \ell) = O(\log s)$ uses of Corollary 6.9 in Line 2.

For correctness, assume that A/B is in its reduced form. Since we know $b_1 = 1$, let i be the largest index with the denominator b_i of the convergent a_i/b_i satisfies $b_i < 2^s$. If this is the last convergent, then $A/B = a_i/b_i$ and we are done. Thus, we may assume that $A/B \neq a_i/b_i$.

Since we are given that $|A/B - a/b| < 1/2^{2s+1} < 1/2b^2$, by Lemma 6.5 Item 4, a/b is one of the convergents of A/B. For any $\ell > i$, the ℓ-th convergent a_ℓ/b_ℓ has denominator larger than 2^s. For any $j < i$, from Lemma 6.5 Item 3 and Item 1 we have

$$\left|\frac{A}{B} - \frac{a_j}{b_j}\right| \geq \frac{1}{b_j(b_j + b_{j+1})} > \frac{1}{2 \cdot b_i^2} \geq \frac{1}{2^{2s+1}}.$$

Thus, a/b must be the i-th convergent a_i/b_i. □

6.4 Algorithm for exact-MME over rationals

We now have all the necessary ingredients to describe the algorithm to prove Theorem 6.1.

Algorithm 7: `Exact-MME-Rationals`

Input : A polynomial $f(x_1, \ldots, x_m) \in \mathbb{Q}_{(-1,1)}[\mathbf{x}]$, points $\mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(N)} \in \mathbb{Q}^m_{(-1,1)}$, with all rational numbers provided via the numerator and denominator, and an integer parameter s such that all numerators and denominators of the coefficients of f, coordinates of the points, and evaluations $f(\mathbf{a}^{(i)})$ are at most 2^s.

Output: Integers $b_1, \ldots, b_N$ and $c_1, \ldots, c_N$ such that $f(\mathbf{a}^{(i)}) = b_i/c_i$ for all $i \in [N]$.

1 Using the numerators and denominators for the required approximation oracles, run `approximate-MME-Reals` $\left(f, \left\{\mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(N)}\right\}, t = 2s+1\right)$ (Algorithm 5) to obtain integers $(B_1, \ldots, B_N)$ such that

$$\left|f(\mathbf{a}^{(i)}) - \frac{B_i}{2^t}\right| < \frac{1}{2^t} = \frac{1}{2^{2s+1}}.$$

2 **for** $i \in N$ **do**

3 Run `Fast-Rational-Number-Reconstruction` $\left(B_i, 2^{2s+1}, s\right)$ (Algorithm 6) to get b_i, c_i with $|c_i| < 2^s$ such that

$$\left|\frac{B_i}{2^{2s+1}} - \frac{b_i}{c_i}\right| < \frac{1}{2^{2s+1}}.$$

4 **return** $(b_1, \ldots, b_N), (c_1, \ldots, c_N)$.

The correctness of Algorithm 7 is evident from Theorem 5.2 and Lemma 6.10. Theorem 5.2 asserts that Algorithm 5 correctly provides the required approximations for the evaluations, and

Lemma 6.10 asserts that Algorithm 6 reconstructs the correct rational number.

As for running time, given the numerator and denominators, we can build approximation oracles for each rational number with nearly-linear running time. Thus, Line 1 takes $((d^m + Nm) \cdot s)^{1+o(1)}$ time and the loop in Lines 2 to 3 takes $\tilde{O}(N \cdot s)$ time. Thus, the total running time is $((d^m + Nm) \cdot s)^{1+o(1)}$ as claimed.

This completes the proof of Theorem 6.1. □

7 Approximate-MME over complex numbers

In this section, we briefly discuss the extension of Theorem 5.2 to the field of complex numbers. As discussed in the preliminaries, the field constants in this case are given by two approximation oracles, one for the real part of the complex number, and one for the imaginary part. The ideas needed for this extension, on top of the ideas in the proof of Theorem 3.1 are quite standard and were introduced by Kedlaya & Umans **[?]** for designing fast algorithms for MME for finite fields that are not prime. This approach also found a subsequent application in the work of Bhargava et al. **[?]**, again in the context of dealing with non-prime finite fields while designing algorithms for MME. In the interest of keeping this discussion succinct and to avoid repetition, we outline the main steps needed for this generalization, but skip the formal details. The structure of the algorithm closely follows that of Algorithm 5, with some additional care.

As in the proof of Algorithm 5, we first make sure that the number of underlying variables is growing. Next, we round each of the field constants (both the real and the imaginary parts) by rational numbers with denominator 2^k for some sufficiently large integer k to be chosen later. At this point, we have introduced some error (which turns out to be small if k is sufficiently large), but have reduced the problem instance over $\mathbb{C}$ to an instance over $\mathbb{Q}[\omega]$, where ω is a square root of -1. Moreover, all the denominators of the field constants in the problem are of the form 2^k. We now clear out the denominators, as in Algorithm 5, and get an instance of MME where the constants in the problem are from the ring $\mathbb{Z}[\omega]$. At this point, we replace ω in the constants in the input by a new formal variable z, and instead of working over the ring $\mathbb{Z}[\omega]$, we work over the ring $\mathbb{Z}[z]/\langle z^2 + 1\rangle$. Note that this is sufficient, since given a solution to MME over this ring, we can obtain a solution to the original problem by just replacing z by ω. Now, the idea is to just invoke the algorithm for exact MME over integers (Algorithm 4) for this problem instance. However, we cannot quite do that directly since the instance at hand is over $\mathbb{Z}[z]/\langle z^2 + 1\rangle$ and not over $\mathbb{Z}$ as desired. Nevertheless, we proceed as in Algorithm 4 by picking sufficiently many primes $p_1, p_2, \ldots, p_s$ and reducing the problem instance over $\mathbb{Z}[z]/\langle z^2 + 1\rangle$ modulo these primes to obtain instances over the rings $\mathbb{F}_{p_i}[z]/\langle z^2 + 1\rangle$ for every i. In Algorithm 4, we just invoked the result of **[?]** over prime fields at this stage, and then combined the outputs using fast Chinese remaindering. However, in this case, what we have are instances over the finite rings $\mathbb{F}_{p_i}[z]/\langle z^2 + 1\rangle$. But this does not turn out to be an issue as the algorithm of Bhargava et al continues to work over such

rings, and indeed the results and proofs in the [**?**] are stated in this form. One final thing to note is that the small optimizations that we do over the results in [**?**] in Section 3 to make sure that the dependence of the running time on the field size is nearly-linear continues to be true for the extension rings that we have here. Once we have solved all the instances over $\mathbb{F}_{p_i}[z]/\langle z^2+1\rangle$, we can recover the solution over $\mathbb{Z}[z]/\langle z^2+1\rangle$ by an application of fast Chinese remaindering as in Algorithm 4, and an appropriate scaling of these evaluations (again, as in Algorithm 5) gives us approximations of the original evaluations over $\mathbb{C}$. The error analysis and the bound on the running time essentially the same as that in the analysis of Algorithm 5. We skip the rest of the details.

8 Discussion and open problems

We conclude with some open problems.

1. Perhaps the most natural question here is to seek an algebraic algorithm for multivariate multipoint evaluation over general fields, both finite and infinite. Currently, we only know such algebraic algorithms over finite fields of small characteristic [Uma08, BGKM22].

2. The aforementioned question of having an algebraic algorithm for MME is also interesting in the non-uniform setting. For instance, we do not know if the linear transformation given by a multivariate Vandermonde matrices can be computed by an arithmetic circuit of nearly-linear (or even sub-quadratic) size over fields other than finite fields of small characteristic.

3. It would be interesting to have additional applications of these faster algorithms and the ideas therein, beyond the applications already mentioned by Kedlaya and Umans [KU11].

References

[BGGKU22] Vishwas Bhargava, Sumanta Ghosh, Zeyu Guo, Mrinal Kumar, and Chris Umans. *Fast multivariate multipoint evaluation over all finite fields*. In Jelani Nelson, ed., *Proc. 63rd IEEE Symp. on Foundations of Comp. Science (FOCS)*, pages 221–232. 2022. arXiv:2205.00342, eccc:2022/TR22-063.

[BGKM22] Vishwas Bhargava, Sumanta Ghosh, Mrinal Kumar, and Chandra Kanta Mohapatra. *Fast, algebraic multivariate multipoint evaluation in small characteristic and applications*. In Stefano Leonardi and Anupam Gupta, eds., *Proc. 54th ACM Symp. on Theory of Computing (STOC)*, pages 403–415. 2022. arXiv:2111.07572, eccc:2021/TR21-162.

[BM74] Allan Borodin and Robert Thomas Moenck. *Fast modular transforms*. J. Comput. Syst. Sci., 8(3):366–386, 1974.

[GG13] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, 3 edition, 2013.

[Had96] JACQUES S. HADAMARD. *Sur la distribution des zéros de la fonction $\zeta(s)$ et ses conséquences arithmétiques*. Bulletin de la Société Mathématique de France, 24:199–220, 1896.

[KU08] KIRAN S. KEDLAYA and CHRISTOPHER UMANS. *Fast modular composition in any characteristic*. In R. RAVI, ed., *Proc. 49th IEEE Symp. on Foundations of Comp. Science (FOCS)*, pages 146–155. 2008.

[KU11] ———. *Fast polynomial factorization and modular composition*. SIAM J. Comput., 40(6):1767–1802, 2011. (Preliminary version in *40th STOC*, 2008 and *49th FOCS*, 2008).

[LVP97] CHARLES JEAN DE LA VALLÉE POUSSIN. *Recherches analytiques sur la théorie des nombres premiers (French) [Analytical research on the theory of prime numbers]*, volume 1–5. Hayez, 1897.

[Möl08] NIELS MÖLLER. *On Schönhage's algorithm and subquadratic integer GCD computation*. Math. Comput., 77(261):589–607, 2008.

[Mor21] GUILLAUME MOROZ. *New data structure for univariate polynomial approximation and applications to root isolation, numerical multipoint evaluation, and other problems*. In NISHEETH VISHNOI, ed., *Proc. 62nd IEEE Symp. on Foundations of Comp. Science (FOCS)*, pages 1090–1099. 2021. arXiv:2106.02505.

[NZ04] MICHAEL NÜSKEN and MARTIN ZIEGLER. *Fast multipoint evaluation of bivariate polynomials*. In SUSANNE ALBERS and TOMASZ RADZIK, eds., *Proc. 12th Annual European Symp. of Algorithms (ESA)*, volume 3221 of *LNCS*, pages 544–555. Springer, 2004. arXiv:cs/0403022.

[Sch80] WOLFGANG M. SCHMIDT. *Diophantine Approximation*, volume 785 of *LNM*. Springer, 1980.

[Uma08] CHRISTOPHER UMANS. *Fast polynomial factorization and modular composition in small characteristic*. In CYNTHIA DWORK, ed., *Proc. 40th ACM Symp. on Theory of Computing (STOC)*, pages 481–490. 2008.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il