COMPUTING GRÖBNER BASES FOR VANISHING IDEALS OF FINITE SETS OF POINTS

JEFFREY B. FARR AND SHUHONG GAO

Abstract. We present an algorithm that incrementally computes a Gröbner basis for the vanishing ideal of any finite set of points in an affine space under any monomial order, and we apply this algorithm to polynomial interpolation in multiple variables. For the case of distinct points, the algorithm is a natural generalization of Newton's interpolation for univariate polynomials. The time complexity in the worst case is exponential in term of the number of variables. Computational evidence suggests, however, that it compares favorably with two known algorithms when the number of variables is small relative to the number of points. We also present a preprocessing technique that significantly enhances the performance of all the algorithms considered. For points with nontrivial multiplicities (defined by delta sets), we adapt our algorithm to compute the vanishing ideal via Taylor expansions.

1. Introduction

Let $\mathbb{F}$ be a field, and suppose $P_1, \ldots, P_n$ are distinct points in $\mathbb{F}^m$. The set of polynomials in $\mathbb{F}[x_1, \ldots, x_m]$ that evaluate to 0 at each point P_i form a zero-dimensional ideal, called the vanishing ideal of the points. The problem we consider is to compute the reduced Gröbner basis for the vanishing ideal of any finite set of points, under any given monomial order. A polynomial time algorithm for this problem was first given by Buchberger and Möller (1982) [3], and significantly improved by Marinari, Möller and Mora (1993) [13], and Abbott, Bigatti, Kreuzer and Robbiano (2000) [1]. These algorithms perform Gauss elimination on a generalized Vandermonde matrix and have a polynomial time complexity. Recently, O'Keeffe and Fitzpatrick (2002) [8] studied this problem from a coding theory point of view. They present an algorithm that is exponential in the number of variables, and the Gröbner basis which they compute is not reduced.

We present an alternate method that is a generalization of Newton's interpolation for univariate polynomials. Our algorithm is similar to O'Keeffe's and Fitzpatrick's approach but computes the reduced Gröbner basis. Even though the time complexity of our algorithm is still exponential, its practical performance improves upon both O'Keeffe's and Fitzpatrick's algorithm and the linear algebra approach mentioned above when the number of variables is relatively small compared to the

Date: January 3, 2004.

This work was supported in part by National Science Foundation (NSF) under Grants DMS9970637 and DMS0302549, National Security Agency (NSA) under Grant MDA904-02-1-0067, the DoD Multidisciplinary University Research Initiative (MURI) program administered by the Office of Naval Research (ONR) under Grant N00014-00-1-0565.

number of points. We provide running time comparisons based on computer experiments for various monomial orders. We also present a preprocessing technique that significantly enhances the performance of our algorithm, the O'Keeffe-Fitzpatrick algorithm and, surprisingly, even the Gauss elimination algorithms.

The rest of the paper is organized as follows. In Section 2, we present our algorithm first for the simple case when the points are distinct. Section 3 deals with an application of our solution to the problem of multivariate polynomial interpolation. Section 4 gives running time comparisons, and a preprocessing technique for sorting the points is presented. Finally, Section 5 shows how to efficiently compute Taylor expansions, which are then used in Section 6 to handle the case for points with multiplicity.

2. Distinct points

In this section we present a solution to the problem of computing a Gröbner basis for the vanishing ideal of a finite set of distinct points. The reader is referred to [2, 6, 12] for introduction to Gröbner bases.

We fix an arbitrary monomial order on the polynomial ring over a field $\mathbb{F}$ with m variables, $\mathbb{F}[x_1, \dots, x_m]$. For any subset $V \subseteq \mathbb{F}^m$, define

$$\mathbf{I}(V) = \{f \in \mathbb{F}[x_1, \dots, x_m] : f(P) = 0, \text{ for all } P \in V\},$$

the vanishing ideal of V. If $V = \{P_1, \dots, P_n\}$, $I(V)$ is also written as $\mathbf{I}(P_1, \dots, P_n)$. For any polynomials $g_1, \dots, g_s \in \mathbb{F}[x_1, \dots, x_m]$, define

$$\mathcal{B}(g_1, \dots, g_s) = \{\mathbf{x}^\alpha : \alpha \in \mathbb{N}^m \text{ and } \mathrm{LT}(g_i) \nmid \mathbf{x}^\alpha, 1 \le i \le s\},$$

where $\mathbb{N} = \{0, 1, 2, 3, \dots\}$, $\mathrm{LT}(g)$ is the leading term of g, and $\mathbf{x}^\alpha = x_1^{\alpha_1} \cdots x_m^{\alpha_m}$ for $\alpha = (\alpha_1, \dots, \alpha_m)$.

Lemma 1. *For $g_1, \dots, g_s \in \mathbf{I} = \mathbf{I}(P_1, \dots, P_n)$, $\{g_1, \dots, g_s\}$ is a Gröbner basis for $\mathbf{I}$ if and only if $|\mathcal{B}(g_1, \dots, g_s)| = n$.*

Proof: Observe that the elements of $\mathcal{B}(\mathbf{I})$ form a basis for the quotient ring $\mathbb{F}[x_1, \dots, x_m]/\mathbf{I}$ viewed as a vector space over $\mathbb{F}$ (see Section 3 in [5]), which implies that $g_1, \dots, g_s \in \mathbf{I}$ form a Gröbner basis iff $\mathcal{B}(g_1, \dots, g_s) = \mathcal{B}(\mathbf{I})$ (see Lemma 3.8 in [13]). By an interpolation method one can show that $\dim \mathbb{F}[x_1, \dots, x_m]/\mathbf{I} = n$. Hence the lemma follows. $\square$

Our algorithm is based on the following lemma.

Lemma 2. *Suppose $G = \{g_1, \dots, g_s\}$ is a Gröbner basis for $\mathbf{I}(V)$, for a finite set $V \subset \mathbb{F}^m$. For a point $P = (a_1, \dots, a_m) \notin V$, let g_i denote the polynomial in G with smallest leading term such that $g_i(P) \neq 0$, and define*

$$\begin{aligned} \tilde{g}_j &:= g_j - \frac{g_j(P)}{g_i(P)} \cdot g_i, \qquad j \neq i, \text{ and} \\ g_{ik} &:= (x_k - a_k) \cdot g_i, \qquad 1 \le k \le m. \end{aligned}$$

Then

$$\widetilde{G} = \{\tilde{g}_1, \dots, \tilde{g}_{i-1}, \tilde{g}_{i+1}, \dots, \tilde{g}_s, g_{i1}, \dots, g_{im}\}$$

is a Gröbner basis for $\mathbf{I}(V \cup \{P\})$.

Proof: By Lemma 1, at least one polynomial in G must be nonzero when evaluated at P; hence, g_i exists.

Certainly, $\widetilde{G} \subseteq \mathbf{I}(V \cup \{P\})$ as the new and modified polynomials evaluate to zero at all points in $V \cup \{P\}$. Denote $\mathrm{LT}(g_i)$ by $\mathbf{x}^\alpha$. We claim that

$$\mathcal{B}(\widetilde{G}) = \mathcal{B}(G) \cup \{\mathbf{x}^\alpha\}. \tag{1}$$

By the choice of i, $\mathrm{LT}(\tilde{g}_j) = \mathrm{LT}(g_j)$, for all $j \neq i$. Also, since g_i was replaced in $\widetilde{G}$ by $g_{i1}, g_{i2}, \ldots, g_{im}$, whose leading terms are $\mathbf{x}^\alpha x_1$, $\mathbf{x}^\alpha x_2$, $\ldots$, $\mathbf{x}^\alpha x_m$, we know that $\mathbf{x}^\alpha$ is the only monomial not in $\mathcal{B}(\mathbf{I}(V))$ that is in $\mathcal{B}(\mathbf{I}(V \cup \{P\}))$. Thus, (1) is satisfied, and $|\mathcal{B}(\widetilde{G})| = |\mathcal{B}(G)| + 1$. Since G is a Gröbner basis for $\mathbf{I}(V)$, we have $|\mathcal{B}(G)| = |V|$, and the conclusion follows from Lemma 1. □

Notice that for some of the g_{ik}, $\mathrm{LT}(g_{ik})$ may be divisible by the leading term of another polynomial in $\widetilde{G}$. In such a case, g_{ik} may be omitted from $\widetilde{G}$ as $\widetilde{G} \setminus \{g_{ik}\}$ still has the same set of leading terms. In fact, we can check for this property before computing g_{ik} so that we save ourselves needless computation. In so doing, we also guarantee that the resulting $\widetilde{G}$ is a minimal Gröbner for $\mathbf{I}(V \cup \{P\})$.

We must, however, be even more careful if we wish to compute the (unique) reduced Gröbner basis. Notice that the reduction of any polynomial $g \in G$ with respect to $G \setminus \{g\}$ requires the use of only those polynomials in G which have leading term smaller than $\mathrm{LT}(g)$. Thus, it is easily seen that the $\tilde{g}_j$, $j \neq i$, are already in normal form since G was reduced to begin with. Any of the g_{ik}, though, may need to be reduced. If upon computing g_{ik} we immediately reduce it with respect to the "current" G (before computing the remaining $g_{i(k+1)}, \ldots, g_{im}$), then we must recompute the normal form of g_{ik} if one of the later $g_{ik'}$ is smaller than g_{ik}. To circumvent this situation, we order the variables so that $x_1 < x_2 < \cdots < x_m$. Thus, in Algorithm 1 G is always stored in such a way that the leading terms of all of its polynomials are in increasing order; hence, each g_{ik} need only be reduced once. In the algorithm below, $Normal(h, G)$ denotes the unique remainder of h when reduced by polynomials in G.

Lemma 2 and the above remarks imply the following theorem.

Theorem 3. *For a finite set $V \subseteq \mathbb{F}^m$ and a given monomial order, Algorithm 1 returns the reduced Gröbner basis for $\mathbf{I}(V)$.*

3. Polynomial Interpolation

Suppose we have points $P_1, \ldots, P_n \in \mathbb{F}^m$ and n values $r_1, \ldots, r_n \in \mathbb{F}$. The (multivariate if $m > 1$) interpolation problem is to find a "smallest" polynomial $f \in \mathbb{F}[x_1, \ldots, x_m]$ so that

$$f(P_i) = r_i, \quad 1 \leq i \leq n. \tag{2}$$

Multivariate polynomial interpolation has been extensively studied in the past 30 years; see Gasca and Sauer [10] for a recent survey of the literature.

We show here that there is a simple relation between multivariate polynomial interpolation problem and Gröbner bases. Specifically, the monomial basis of $\mathbf{I}(P_1, \ldots, P_n)$ is a unique interpolation space, and the interpolating polynomial

Algorithm 1

```
Input: $P_1, P_2, \ldots, P_n \in \mathbb{F}^m$, and a monomial order.
Output: $G$, the reduced Gröbner basis for $\mathbf{I}(P_1, \ldots, P_n)$, in increasing order.

/* Initialization */
$G := \{1\}$;          /* the $i$th polynomial in $G$ is denoted $g_i$ */
Order the variables so that $x_1 < x_2 < \cdots < x_m$;

FOR $k$ from 1 to $n$ DO
  Find the smallest $i$ so that $g_i(P_k) \neq 0$;
  FOR $j$ from $i+1$ to $|G|$ DO   $g_j := g_j - \frac{g_j(P_k)}{g_i(P_k)} \cdot g_i$;   END FOR;
  $G := G \setminus \{g_i\}$;
  FOR $j$ from 1 to $m$ DO
    IF $x_j \cdot \mathrm{LT}(g_i)$ not divisible by any LT of $G$ THEN
      Compute $h := Normal((x_j - a_j) \cdot g_i, G)$;
      Insert $h$ (in order) into $G$;
    END IF;
  END FOR;
END FOR;

RETURN $G$.
```

may be found by computing the reduced Gröbner basis for the vanishing ideal of the augmented points $(P_1, r_1), \ldots, (P_n, r_n)$.

Theorem 4. *Fix any monomial order on $\mathbb{F}[x_1, \ldots, x_m]$, and let G be the reduced Gröbner basis for $\mathbf{I} = \mathbf{I}(P_1, \ldots, P_n)$ and $\mathcal{B} = \mathcal{B}(\mathbf{I}) = \{\mathbf{x}^{\alpha_1}, \ldots, \mathbf{x}^{\alpha_n}\}$, the corresponding monomial basis.*

(i) *For any $r_1, \ldots, r_n \in \mathbb{F}$, there is a unique $f \in Span_{\mathbb{F}}(\mathcal{B})$ satisfying (2).*

(ii) *Introduce a variable z together with the unique elimination order for z in $\mathbb{F}[x_1, \ldots, x_m, z]$ that extends the given monomial order on $\mathbb{F}[x_1, \ldots, x_m]$. Then the reduced Gröbner basis for*

$$\mathbf{I}((P_1, r_1), \ldots, (P_n, r_n)),$$

is of the form $G \cup \{z - f\}$, where f is the unique polynomial in (i).

Proof: Part (i) is standard; we show (ii). For the polynomial $f \in \mathbb{F}[x_1, \ldots, x_m]$ satisfying (2), we have $z - f \in \mathbf{I}((P_1, r_1), \ldots, (P_n, r_n))$. Note for any elimination order for z in $\mathbb{F}[x_1, \ldots, x_m, z]$, z is the leading term in $z - f$. Hence

$$\mathbb{F}[x_1, \ldots, x_m, z]/\mathbf{I}((P_1, r_1), \ldots, (P_n, r_n))$$

is isomorphic (as a ring) to $\mathbb{F}[x_1, \ldots, x_m]/\mathbf{I}(P_1, \ldots, P_n)$, both have $\mathcal{B}(\mathbf{I}(P_1, \ldots, P_n))$ as their monomial basis. Therefore, $G \cup \{z - f\}$ is equal to the unique reduced Gröbner basis for $\mathbf{I}((P_1, r_1), \ldots, (P_n, r_n))$. $\square$

By the above theorem, the interpolation polynomial f can be found by simply applying Algorithm 1 to the augmented points $(P_1, r_1), \ldots, (P_n, r_n)$ with an elimination order for z. It will find f and a Gröbner basis for $\mathbf{I}(P_1, \ldots, P_n)$ simultaneously!

4. Time Complexity

4.1. **The cost of reduction.** All the steps in Algorithm 1 are straightforward except the reduction step in line 14. We use the standard long-division technique, sometimes called Buchberger reduction. This reduction has a worst-case time complexity that may be exponential in the number m of variables. It is possible to make this step polynomial time by using the border-basis reduction technique introduced in [7]. To compute a border Gröbner basis, we simply replace "not divisible by" in line 13 with "not equal to", and we replace the reduction step in line 14 by a simpler border-basis reduction. The border Gröbner basis computed, however, is not a reduced Gröbner basis and is, in general, quite large. For example, the reduced Gröbner basis for the vanishing ideal of a random set of 500 points from ${\mathbb{F}_2}^{10}$ under *lex* order usually contains around 100 polynomials, while the border basis typically contains over 2000! Hence the average running time of Algorithm 1 using border-basis reduction is much worse than the original. Additionally, due to the size of the border basis, memory concerns become an issue. For these reasons we ignore the theoretical "improvements" that border-basis reduction provides. In the time comparisons below, Algorithm 1 is implemented with the theoretically worse, but practically better, Buchberger reduction.

4.2. **Comparison with current methods.** As we mentioned earlier, the methods in Buchberger and Möller (1982) [3], Marinari, Möller and Mora (1993) [13], and Abbott, Bigatti, Kreuzer and Robbiano (2000) [1] are based on Gauss elimination and have a polynomial time complexity $O(n^3m)$. We compare our Algorithm 1 particularly with the algorithm (which we designate MMM) of Marinari, Möller and Mora [13].

We denote the algorithm of O'Keeffe and Fitzpatrick [8] by O'K-F. The Gröbner basis found via this method is minimal in the sense that the number of polynomials in the basis is the smallest, but the length of the polynomials computed may grow exponentially in the number m of variables. Hence it has an exponential time complexity. For example, for 200 random points in $\mathbb{F}_5^{10}$, the largest polynomial in O'K-F's Gröbner basis typically has roughly 300 terms for *glex* order, and roughly 1500 terms for pure *lex* order. So, most of the computing time in O'K-F is taken up with dealing with large polynomials, and most of the time in Algorithm 1 involves the reduction step, *i.e.*, computing $Normal(g_{ij}, G)$.

Tables 1 - 3 present running times for Algorithm 1, MMM and O'K-F for various point sets. To highlight the significance of the dimension, we have chosen three vector spaces over $\mathbb{F}_2$ and three other vector spaces of relatively the same size, but of low dimension; *e.g.*, $\mathbb{F}_2^{10}$ has approximately the same number of points as $\mathbb{F}_{11}^3$. The times are the average running times (in seconds) for randomly chosen point sets from the specified vector space (based on 100 experiments for $n = 250$, 10 experiments for $n = 500, 1000$). The algorithms were implemented in MAGMA version 2.8 and run on a SUN Blade 1000, 750 MHz Ultra3 CPU, 512 MB RAM. Entries marked *** indicate that memory was exhausted.

q	m	MMM		Algorithm 1		O'Keeffe-Fitzpatrick	
		glex	*lex*	*glex*	*lex*	*glex*	*lex*
2	10	11.56	4.37	9.38	3.18	13.08	24.45
2	15	39.85	9.28	42.49	19.18	41.32	61.37
2	20	110.08	13.72	152.06	44.18	106.99	93.52
11	3	9.60	5.21	4.25	1.08	3.83	2.40
31	3	11.20	5.64	5.10	0.988	4.57	1.44
101	3	11.57	5.53	5.31	0.747	4.75	0.833
1009	3	12.51	6.41	5.70	0.477	5.03	0.464

Table 1. Average running times for 250 random points from $\mathbb{F}_q^m$ (based on 100 experiments)

q	m	MMM		Algorithm 1		O'Keeffe-Fitzpatrick	
		glex	*lex*	*glex*	*lex*	*glex*	*lex*
2	10	52.75	24.40	36.98	13.00	81.05	915.41
2	15	293.24	50.50	311.14	96.82	301.93	1094.0
2	20	553.61	77.29	677.23	308.59	608.32	2091.4
11	3	64.18	39.10	21.33	5.50	19.41	22.12
31	3	84.32	41.35	33.25	5.43	29.60	10.16
101	3	86.30	42.44	35.30	3.61	31.23	4.77
1009	3	90.24	46.58	36.27	2.16	31.61	2.10

Table 2. Average running times for 500 random points from $\mathbb{F}_q^m$ (based on 10 experiments)

q	m	MMM		Algorithm 1		O'Keeffe-Fitzpatrick	
		glex	*lex*	*glex*	*lex*	*glex*	*lex*
2	12	727.46	227.04	604.10	197.08	1269.49	***
2	15	1633.0	328.63	1659.6	545.30	2394.4	***
2	20	4442.8	508.98	5606.9	2624.4	4872.0	***
31	3	752.16	331.43	237.34	31.86	213.37	83.06
101	3	762.07	356.20	243.21	20.96	218.34	33.51
1009	3	724.02	362.60	256.70	11.10	220.14	11.88

Table 3. Average running times for 1000 random points from $\mathbb{F}_q^m$ (based on 10 experiments)

The tables seem to indicate that Algorithm 1 has a decided advantage over MMM provided the dimension is small relative to the number n of points. If m is much larger, then the advantage swings to MMM; however, Algorithm 1 seems to catch up in the $m = 15$ cases as n increases. The number of points would have to be much greater than 1000 before Algorithm 1 could compete in the $m = 20$ case.

Interestingly, if the dimension m and the number of points n are fixed and the field size is allowed to grow, then the running time for Algorithm 1 and O'K-F under *lex* order actually *decreases*. The reason is that the Gröbner basis polynomials actually become simpler and the reduction by these polynomials is faster. The MMM algorithm experiences no such speedup, although the increase in its running time is mild.

4.3. **Sorting the Points.** It is somewhat surprising that a clever ordering of the points can improve the running time of Algorithm 1. This improvement is more or less significant depending on both the chosen monomial order and the geometric structure of the points. Additionally, this special ordering does improve the running time of the Fitzpatrick-O'Keeffe algorithm, drastically in some cases. Even more surprising is the fact that this ordering also speeds up the MMM algorithm.

The details of this ordering, motivated by [9], are quite simple. If $x_1 < x_2 < \cdots < x_m$, then group the points first according to the x_1-coordinate; these groups are ordered in a nonincreasing order by size. Within each of the groups, repeat the process, but according to the x_2-coordinate. Continue for $x_3, \ldots, x_m$.

Example 1. Under any order with $x_1 < x_2 < x_3$, the set of points

$$\{(4,0,0),(2,1,4),(2,4,0),(3,0,1),(2,1,3),(1,3,4),(2,4,3),(2,4,2),(1,0,2)\}$$

is reordered as

$$\{(2,4,0),(2,4,2),(2,4,3),(2,1,3),(2,1,4),(1,3,4),(1,0,2),(3,0,1),(4,0,0)\}.$$

Essentially, this sorting decreases the amount of reduction that Algorithm 1 does. Further, the Fitzpatrick-O'Keeffe algorithm, though it does not involve reduction, is also helped since the Gröbner basis remains comparatively small. In the MMM algorithm, reordering the points corresponds to a favorable reordering of the columns in a matrix before Gauss elimination is applied.

Gröbner bases under *lex* order experience the greatest speedup since they typically require the most reduction and are prone to exponential growth without reduction, and Gröbner bases under a *glex* order with points from a low-dimensional vector space experience little to no speedup. A comparison of Tables 4 - 6 below with Tables 1 - 3 indicates the sizable impact that reordering gives.

q	m	MMM		Algorithm 1		O'Keeffe-Fitzpatrick	
		glex	*lex*	*glex*	*lex*	*glex*	*lex*
2	10	7.78	2.72	6.60	1.84	7.17	2.22
2	15	32.71	6.24	35.65	9.17	31.58	7.14
2	20	98.97	9.26	139.66	24.05	92.93	11.28
11	3	8.11	2.80	3.96	0.944	3.43	1.01
31	3	11.15	3.72	5.10	0.932	4.47	0.950
101	3	11.63	4.18	5.31	0.721	4.71	0.704
1009	3	12.52	6.11	5.70	0.469	5.02	0.462

TABLE 4. Average running times for 250 random points (sorted) from $\mathbb{F}_q^m$ (based on 100 experiments)

q	m	MMM		Algorithm 1		O'Keeffe-Fitzpatrick	
		glex	*lex*	*glex*	*lex*	*glex*	*lex*
2	10	26.50	9.48	21.83	5.76	29.26	9.12
2	15	233.32	27.77	267.23	30.50	222.91	46.37
2	20	466.00	44.26	586.64	142.60	475.94	83.39
11	3	44.71	14.25	19.63	4.50	17.18	5.05
31	3	83.54	21.97	33.77	4.87	28.65	4.97
101	3	86.83	27.14	35.63	3.55	30.51	3.52
1009	3	90.17	41.73	36.16	2.17	31.43	2.04

TABLE 5. Average running times for 500 random points (sorted) from $\mathbb{F}_q^m$ (based on 10 experiments)

q	m	MMM		Algorithm 1		O'Keeffe-Fitzpatrick	
		glex	*lex*	*glex*	*lex*	*glex*	*lex*
2	12	429.45	79.80	416.59	74.34	600.22	152.30
2	15	1200.7	140.15	1306.0	189.94	1481.3	338.86
2	20	3763.5	236.67	5045.0	924.12	3821.9	701.99
31	3	733.93	143.02	238.92	28.22	201.84	30.02
101	3	759.31	182.38	249.26	20.11	214.26	20.24
1009	3	717.05	292.70	254.09	10.62	217.55	10.46

TABLE 6. Average running times for 1000 random points (sorted) from $\mathbb{F}_q^m$ (based on 10 experiments)

5. COMPUTING TAYLOR EXPANSIONS

In this section we describe in detail how to compute the Taylor expansion modulo an ideal I. We use this material in the next section to strengthen Algorithm 1 by allowing points with multiplicities to be considered.

Let $\mathbf{v} = (v_1, \ldots, v_m) \in \mathbb{Z}^m$. We define a differential operator $D^{\mathbf{v}}$ by

$$D^{\mathbf{v}} = \frac{1}{v_1! \cdots v_m!} \cdot \frac{\partial^{v_1+\cdots+v_m}}{\partial x_1{}^{v_1} \cdots \partial x_m{}^{v_m}}.$$

We note that $D^{\mathbf{v}}$ is a linear map on functions with the m variables $x_1, \ldots, x_m$. Let $P \in \mathbb{F}^m$ and f be any function on $x_1, \ldots, x_m$. We employ the notation

$$[D^{\mathbf{v}} f](P) = D^{\mathbf{v}} f|_{\mathbf{x}=P}, \tag{3}$$

where $P = (a_1, \ldots, a_m) \in \mathbb{F}^m$. Then, under reasonable conditions (analytic or algebraic) on f, we have

$$f(\mathbf{x} + P) = \sum_{\mathbf{v} \in \mathbb{N}^m} [D^{\mathbf{v}} f](P) \cdot \mathbf{x}^{\mathbf{v}}. \tag{4}$$

We call the right-hand side of (4) the Taylor expansion of f at P, denoted by $T(f, P)$. Note that (4) is equivalent to

$$f(\mathbf{x}) = \sum_{\mathbf{v} \in \mathbb{N}^m} [D^{\mathbf{v}} f](P) \cdot (\mathbf{x} - P)^v = \sum_{\mathbf{v} \in \mathbb{N}^m} [D^{\mathbf{v}} f](P) \cdot (x_1 - a_1)^{v_1} \cdots (x_m - a_m)^{v_m},$$

which is the more typically referred to form of Taylor expansion.

Suppose $\Delta \subseteq \mathbb{N}^m$ is a finite set, referred to as a *delta set* or a *Ferrers diagram*, satisfying the division order; that is, if $\mathbf{v} = (v_1, \ldots, v_m) < \mathbf{u} = (u_1, \ldots, u_m)$ componentwise and $\mathbf{u} \in \Delta$, then $\mathbf{v} \in \Delta$. Define

$$T(f, P, \Delta) = \sum_{\mathbf{v} \in \Delta} [D^{\mathbf{v}} f](P) \cdot \mathbf{x}^{\mathbf{v}}.$$

$T(f, P)$ denotes the full (possibly infinite if f is not a polynomial) Taylor expansion of f, while $T(f, P, \Delta)$ is truncated to consider only those coefficients corresponding to monomials with exponents in Δ.

Let I denote the monomial ideal generated by monomials with exponents in the complement of Δ:

$$I = \langle \mathbf{x}^{\mathbf{v}} : \mathbf{v} \in (\mathbb{N}^m \setminus \Delta) \rangle.$$

So, $\Delta = \mathcal{B}(I)$ regardless of the monomial order placed on $\mathbb{F}[x_1, \ldots, x_m]$. Assuming we have a point P and a polynomial $f \in \mathbb{F}[x_1, \ldots, x_m]$ given by its coefficients (*i.e.*, $f = \sum_{\mathbf{v} \in S} f_{\mathbf{v}} \mathbf{x}^{\mathbf{v}}$, $f_{\mathbf{v}}$ nonzero elements in $\mathbb{F}$, $S \subseteq \mathbb{N}^m$), the goal is to compute the Taylor expansion of f about P modulo I; by Equation (4), this is equivalent to computing $[D^{\mathbf{v}} f](P)$ for $\mathbf{v} \in \Delta = \mathcal{B}(I)$. Since I is a monomial ideal, any terms with exponent not in Δ can simply be ignored.

Also, notice that if $T(g, P, \Delta)$ and $T(h, P, \Delta)$ are the Taylor expansions (having at most $|\Delta|$ terms apiece) of g and h, respectively, then the Taylor expansion of $g \cdot h$ is

$$T(g \cdot h, P, \Delta) \equiv T(g, P, \Delta) \cdot T(h, P, \Delta) \mod I. \tag{5}$$

Again, computing modulo I is free; simply, drop any terms with exponents outside of Δ.

We break the strategy for computing $[D^{\mathbf{v}} f](P)$ into two cases depending on the density of the set S of terms in f with nonzero coefficients. Define Δ_S to be the smallest delta set containing S, and let δ denote the total degree of f. If $|S|/|\Delta_S|$ is small, then S is said to be *sparse*; otherwise, S is *dense*.

In the former case, one can compute the Taylor expansion term-by-term. For any $\mathbf{v} \in S$, $(\mathbf{x} + P)^{\mathbf{v}}$ can be computed via the square-and-multiply method modulo I. These expansions can then be added together to find $T(f, P, \Delta)$. This approach has time complexity $O(m \cdot |S| \cdot \log \delta \cdot |\Delta|^2)$.

We are more interested in the situation in which S does not have a sparse structure. Here, we can do better than the term-by-term approach used in the sparse case. The reason for this is that the expansions for similar terms share some information. For example, the expansions of the terms x^2yz^3 and $x^2y^2z^2$ both are related to the expansion of x^2yz^2; namely, by Equation 5, $T(x^2yz^3, P, \Delta) \equiv T(z, P, \Delta) \cdot T(x^2yz^2, P, \Delta) \mod I$ and $T(x^2y^2z^2, P, \Delta) \equiv T(y, P, \Delta) \cdot T(x^2yz^2, P, \Delta) \mod I$. By accounting for such dependencies, we can save considerable computing time.

Before presenting Algorithm 2 below to compute $T(f, P, \Delta)$, we make two important observations. First, as seen in the simple illustration above, the order in which we compute the term expansions makes a difference. Clearly, computing the expansion of a larger term *before* computing the expansion of a smaller term would be counterproductive. Hence, we start by computing the constant term and building up from there in all m directions.

The second remark is that we do not need to remember the expansion for each term throughout the entire algorithm. In the illustration, after we compute $T(x^2yz^3, P, \Delta)$, $T(x^2y^2z^2, P, \Delta)$ and $T(x^3yz^2, P, \Delta)$, if necessary, we can forget about $T(x^2yz^2, P, \Delta)$. So, in a sense, we only need to keep track of the term expansions on the "border" (denoted by R in Algorithm 2) of the set of already computed expansions; the other terms may be added to the eventual output and forgotten.

Algorithm 2

```
Input: $P \in \mathbb{F}^m$; a monomial basis $\Delta$ of a monomial ideal $I$;
    $f \in \mathbb{F}[x_1, \ldots, x_m]$ given by its coefficients $f_{\mathbf{v}}$, $\mathbf{v} \in \Delta_S \subseteq \Delta$;
    and a monomial order.
Output: $T(f, P, \Delta)$, the Taylor expansion of $f$ about $P$ modulo $I$.
Variables:
  T– the current Taylor expansion, updated each iteration;
  R– the border of the set
    $\{\mathbf{v}$ : the Taylor expansion of $(x+P)^{\mathbf{v}}$ is already computed$\}$;

/* Initialization */
$R := \{(0, \ldots, 0)\}$;
$T_{\mathbf{0}} := f(P)$;
$T := T_{\mathbf{0}}$;

/* Main */
WHILE $R \neq \varnothing$ DO
  $R_1 = \varnothing$;
  FOR $\mathbf{v} \in R$ DO
    FOR $k$ from 1 to $m$ DO
      $\bar{\mathbf{v}} := \mathbf{v} + e_k$;            /* $e_k$ is the kth unit vector */
      IF $(\bar{\mathbf{v}} \in \Delta_S)$ & $(\bar{\mathbf{v}} \notin R_1)$ THEN
        Append$(\bar{\mathbf{v}}, R_1)$;
        $T_{\bar{\mathbf{v}}} := T_{\mathbf{v}} \cdot (x_k + a_k) \mod I$;
        $T := T + f_{\bar{\mathbf{v}}} \cdot T_{\bar{\mathbf{v}}}$;
      END IF;
    END FOR;
  END FOR;
  $R := R_1$;
END WHILE;

RETURN $T$.
```

TABLE 7. Algorithm for computing Taylor expansions modulo a delta set

Algorithm 2 has time complexity $O(m \cdot |\Delta_S|^2)$. So unless

$$\frac{|S|}{|\Delta_S|} < \frac{|\Delta_S|}{|\Delta|^2 \cdot \log \delta},$$

f is dense and Algorithm 2 outperforms the term-by-term square-and-multiply method.

6. Points with Multiplicities

We now consider the case in which some points in the vanishing set have multiplicity. What is meant by the multiplicity of a solution of a multivariate polynomial varies depending on the context and the author. Marinari, Möller and Mora describe several notions of a general *algebraic* multiplicity [14], one of which is described in detail in [13]. In this paper we use a slightly restricted (but easier to present) form of this latter definition, studied by Cerlienco and Mureddu [4].

For any nonzero polynomial $f \in \mathbb{F}[x_1, \ldots, x_m]$, a point $P \in \mathbb{F}^m$ and a delta set $\Delta \subset \mathbb{N}^m$, P is said to have *multiplicity* Δ if the truncated Taylor expansion of f at P is zero; that is,

$$T(f, P, \Delta) = \sum_{\mathbf{v} \in \Delta} [D^{\mathbf{v}} f](P) \cdot \mathbf{x}^{\mathbf{v}} = 0,$$

where $[D^{\mathbf{v}} f](P)$ is as defined in Equation (3). We assume that Δ is the largest possible delta set satisfying these conditions.

A somewhat more common definition of multiplicity, called *arithmetic* multiplicity, is as follows. A solution P of a polynomial $f \in \mathbb{F}[x_1, \ldots, x_m] = \mathbb{F}[\mathbf{x}]$ is said to have multiplicity m_0 if $[D^{\mathbf{v}} f](P) = 0$ whenever $v_1 + v_2 + \cdots + v_m < m_0$. In terms of the algebraic definition, this implies that the multiplicity set is restricted to a triangular shape. Thus, it is obvious that the algebraic definition of multiplicity subsumes the arithmetic.

With the former definition of multiplicity in mind, we present a generalization of Algorithm 1 that computes the vanishing ideal of a set of points $\{P_1, \ldots, P_n\}$, each having multiplicity defined by the sets $\Delta_1, \ldots, \Delta_n$. Denote this ideal by

$$\mathbf{I}\left((P_1, \Delta_1), \ldots, (P_n, \Delta_n)\right) = \{f \in \mathbb{F}[x_1, \ldots, x_m] : T(f, P_i, \Delta_i) = 0, \quad 1 \leq i \leq n\}.$$

Since the Δ_i's are delta sets, one can show that this set is indeed an ideal in $\mathbb{F}[x_1, \ldots, x_m]$. (We should mention that this is not true if Δ_i's are not all delta sets.)

Algorithm 3 varies from Algorithm 1 in the following ways. First, instead of evaluating a polynomial f at P_i, we need to compute the truncated Taylor expansion $T(f, P_i, \Delta_i)$ using Algorithm 2; we denote the set of these expansions by $\mathcal{T}$. It is important that each Δ_i is ordered according to the division order to ensure that these Taylor expansions may be computed efficiently and to ensure that $\mathbf{I}\left((P_1, \Delta_1), \ldots, (P_n, \Delta_n)\right)$ is actually an ideal. That is, order Δ_i in such a way that no element divides any previous element in Δ_i.

Secondly, we note that Algorithm 3 is also an iterative method; in fact, not only does the algorithm build the Gröbner basis for the vanishing ideal "one point at a time" but it also builds it "one multiplicity at a time." That is, when a new point is introduced, the algorithm updates the Gröbner basis by stepping through the corresponding multiplicity set element by element; this fact is seen in Algorithm 3 by noticing that the FOR loop of line 12 is inside the FOR loop of line 10. Of course if each multiplicity set is trivial ($|\Delta_i| = 1$), then Algorithm 3 is equivalent to Algorithm 1.

Algorithm 3

```
 1  Input: $P_1, \dots, P_n \in \mathbb{F}^m$; $\Delta_1, \dots, \Delta_n \subset \mathbb{N}^m$; and a monomial order.
 2  Output: $G$, the reduced Gröbner basis for $\mathbf{I}((P_1, \Delta_1), \dots, (P_n, \Delta_n))$,
            in increasing order.
 3
 4  /* Initialization */
 5  $G := \{1\}$;      /* $g_i$ is the ith polynomial in $G$, in increasing order */
 6  Order the variables so that $x_1 < x_2 < \cdots < x_m$;
 7  Order the elements in each $\Delta_k$ in nondecreasing order under the
            division order;
 8
 9  /* Main */
10  FOR $k$ from 1 to $n$ DO
11    Compute $\mathcal{T} = \{T_j = T(g_j, P_k, \Delta_k) : g_j \in G\}$, the set of
          (truncated) Taylor expansions;
12    FOR $\mathbf{v}$ in $\Delta_k$ DO
13      Find the smallest $i$ so that coeff$(T_i, \mathbf{x}^{\mathbf{v}}) \neq 0$;
14      FOR $j$ from $i+1$ to $|G|$ DO
15        $\delta :=$ coeff$(T_j, \mathbf{x}^{\mathbf{v}})$ / coeff$(T_i, \mathbf{x}^{\mathbf{v}})$;
16        $g_j = g_j - \delta \cdot g_i$;
17        $T_j = T_j - \delta \cdot T_i$;
18      END FOR;
19      $G := G \setminus \{g_i\}$ and $\mathcal{T} := \mathcal{T} \setminus \{T_i\}$;
20      FOR $j$ from 1 to $m$ DO
21        IF $x_j \cdot \mathrm{LT}(g_i)$ not divisible by any LT of $G$ THEN
22          Compute $h := Normal((x_j - a_j) \cdot g_i, G)$;
23          $T_h := x_j \cdot T_i$    (truncated);
24          Insert (in order) $h$ into $G$ and $T_h$ into $\mathcal{T}$;
25        END IF;
26      END FOR;
27    END FOR;
28  END FOR;
29
30  RETURN $G$.
```

TABLE 8. Algorithm for computing the reduced Gröbner basis for the vanishing ideal of a set of points with multiplicities

The following analogue to Lemma 1 is necessary to establish the correctness of Algorithm 3. We omit a formal proof of the correctness of Algorithm 3, noting only that the key step that established Algorithm 1 is the same for this algorithm. Namely, at each step in the algorithm, we add exactly one element from a multiplicity set and exactly one element to the monomial basis. This ensures that our basis G is always Gröbner.

Lemma 5. *Fix a monomial order on* $\mathbb{F}[\mathbf{x}]$*, and let* $V = \{(P_1, \Delta_1), \ldots, (P_n, \Delta_n)\}$ *where* $P_i \in \mathbb{F}^m$ *are distinct points and* $\Delta_i \subset \mathbb{N}^m$ *are delta sets. Then* $\{g_1, \ldots, g_s\} \subset \mathbf{I}(V)$ *is a Gröbner basis for* $\mathbf{I}$ *if and only if* $|\mathcal{B}(g_1, \ldots, g_s)| = \sum_{j=1}^{n} |\Delta_j|$.

Proof: We know that $g_1, \ldots, g_s \in \mathbf{I}(V)$ form a Gröbner basis iff $|\mathcal{B}(g_1, \ldots, g_s)| = \dim \mathbb{F}[x_1, \ldots, x_m]/\mathbf{I}(V)$; see Lemma 3.8 in [13]. We just need to show that the latter has dimension equal to $\sum_{j=1}^{n} |\Delta_j|$. To see this, let $I_j = \mathbf{I}(P_j, \Delta_j)$, the vanishing ideal of P_j with multiplicity Δ_j. Then $\mathbf{I}(V) = I_1 \cap \cdots \cap I_n$ and

$$\mathbb{F}[x_1, \ldots, x_m]/\mathbf{I}(V) \cong \bigotimes_{j=1}^{n} \mathbb{F}[x_1, \ldots, x_m]/I_j,$$

as rings over $\mathbb{F}$. Note that $\{\mathbf{x}^\alpha : \alpha \in \Delta_j\}$ form a basis for $\mathbb{F}[x_1, \ldots, x_m]/I_j$ as a vector space over $\mathbb{F}$, so its dimension equal to $|\Delta_j|$. The lemma follows immediately. $\square$

Example 2. Consider the following simple example with two points in $\mathbb{F}_3^2$. Let

$$P_1 = (0,0), \qquad \Delta_1 = \{(0,0),(1,0),(0,1),(2,0),(1,1),(0,2),(3,0)\},$$
$$\text{and} \qquad P_2 = (1,2) \qquad \Delta_2 = \{(0,0),(1,0),(0,1)\}.$$

We assume a *glex* order on $\mathbb{F}_3[x, y]$ with $y > x$.

k	$\mathbf{v}$	i	G	$\mathcal{T}$
1	$(0,0)$	*1*	$\{1\}$	$\{1\}$
1	$(1,0)$	*1*	$\{x, y\}$	$\{x, y\}$
1	$(0,1)$	*1*	$\{y, x^2, xy\}$	$\{y, x^2, xy\}$
1	$(2,0)$	*1*	$\{x^2, xy, y^2\}$	$\{x^2, xy, y^2\}$
1	$(1,1)$	*1*	$\{xy, y^2, x^3\}$	$\{xy, y^2, x^3\}$
1	$(0,2)$	*1*	$\{y^2, x^3, x^2y\}$	$\{y^2, x^3, 0\}$
1	$(3,0)$	*1*	$\{x^3, x^2y, xy^2, y^3\}$	$\{x^3, 0, 0, 0\}$
2	$(0,0)$	*1*	$\{x^2y, xy^2, y^3, x^4\}$	$\{y+x-1, y+x+1, -1, x+1\}$
2	$(1,0)$	*1*	$\{xy^2+x^2y, y^3-x^2y, x^4+x^2y, x^3y-x^2y\}$	$\{-y-x, -y-x, y-x, -x\}$
2	$(0,1)$	*2*	$\{y^3-xy^2+x^2y,\ x^4-xy^2,\ x^3y+xy^2, x^2y^2+xy^2-x^2y\}$	$\{0, -y, y, 0\}$
OUTPUT			$G = \{y^3-xy^2+x^2y, x^3y+x^4, x^2y^2+xy^2-x^2y, x^5-x^4-xy^2-x^2y\}$	

TABLE 9. Results of Algorithm 3 in Example 2

Table 9 shows the results of Algorithm 3 at each iteration after line 13 has been performed. It is easy to verify by hand that the output

$$G = \{y^3 - xy^2 + x^2y, x^3y + x^4, x^2y^2 + xy^2 - x^2y, x^5 - x^4 - xy^2 - x^2y\}$$

is a subset of I, and an examination of $\mathcal{B}(G)$ together with Lemma 5 proves that G is a Gröbner basis for I.

7. Final Remarks

We have presented an algorithm to compute the Gröbner basis for the vanishing ideal of any finite set of affine points over any field. Additionally, we adapt our method to handle the case in which some points have nontrivial multiplicities given by a delta set.

While the method is described for any field, the authors are most interested in applications over finite fields, as many of the examples indicate. Hence, we have left many numerical questions that arise from the use of nonexact arithmetic unstudied. Particularly, the question of the conditioning of our solution may be of interest. The Gauss elimination techniques that are in practice involve Vandermonde matrices which are often ill-conditioned, and our algorithm could produce an advantage in this area. Another numerical issue to deal with is the growth of coefficients when working over the rational number field. Modular methods have been effective in controlling such growth in other Gröbner basis algorithms [1, 11]; we expect that a similar idea would work for our algorithm.

References

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

[2] William W. Adams and Philippe Loustaunau, *An introduction to Gröbner bases*, Graduate Studies in Mathematics, 3, American Mathematical Society, Providence, RI, 1994.

[3] B. Buchberger and H. M. Möller, The construction of multivariate polynomials with preassigned zeros. *Computer algebra, EUROCAM '82*, pp. 24-31, Lecture Notes in Comput. Sci., vol. 144, Springer, Berlin-New York, 1982.

[4] L. Cerlienco and M. Mureddu, From algebraic sets to monomial linear bases by means of combinatorial algorithms, Formal power series and algebraic combinatorics (Montreal, PQ, 1992), *Discrete Math.* **139** (1995), no. 1-3, 73-87.

[5] David Cox, John Little and Donal O'Shea, *Ideals, varieties, and algorithms*, 2nd ed., Undergraduate Texts in Mathematics, Springer-Verlag, New York, 1997.

[6] David Cox, John Little and Donal O'Shea, *Using algebraic geometry*, Graduate Texts in Mathematics, 185, Springer-Verlag, New York, 1998.

[7] J. Faugere, P. Gianni, D. Lazard and T. Mora, Efficient computation of zero-dimensional Gröbner bases by change of ordering, *J. Symbolic Comput.* **16** (1993), 329-344.

[8] Patrick Fitzpatrick and Henry O'Keeffe, Gröbner basis solutions of constrained interpolation problems, Fourth special issue on linear systems and control, *Linear Algebra Appl.* **351/352** (2002), 533-551.

[9] Shuhong Gao, Virgínia M. Rodrigues and Jeffrey Stroomer, Gröbner basis structure of finite sets of points, *preprint.*

[10] Mariano Gasca and Thomas Sauer, Polynomial interpolation in several variables, in Multivariate polynomial interpolation, *Adv. Comput. Math.* **12** (2000), no. 4, 377-410.

[11] J. de Kleine and M. Monagan, A modular method for computing Gröbner bases, *preprint.*

[12] M. Kreuzer and L. Robbiano, *Computational Commutative Algebra 1*, Springer-Verlag, Berlin, 2000.

[13] 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 Engrg. Comm. Comput.* **4** (1993), no. 2, 103-145.

[14] M. G. Marinari, H.M. Möller and T. Mora, On multiplicities in polynomial system solving, *Trans. Amer. Math. Soc.* **348** (1996), no. 8, 3283-3321.

[15] Thomas Sauer, Polynomial interpolation of minimal degree and Gröbner bases, *Gröbner bases and applications* (Linz, 1998), 483-494, London Math. Soc. Lecture Note Ser., vol. 251, Cambridge Univ. Press, Cambridge, 1998.

Department of Mathematical Sciences, Clemson University, Clemson, SC 29634-0975, USA *E-mail address*: {jeffref, sgao}@ces.clemson.edu