Towards a Generalization of Fulton's Intersection Multiplicity Algorithm

Ryan Sandford

Department of Computer Science, The University of Western Ontario, Canada

April 25, 2022

Contributions

- We extend Fulton's bivariate intersection multiplicity algorithm to a partial intersection multiplicity algorithm in the n-variate setting.
- We extend the generalization of Fulton's algorithm to handle points with non-rational coordinates by encoding such points in a zero-dimensional regular chain.
- We implement both versions of the generalizatiom of Fulton's algorithm in MAPLE, combining the latter version with the algorithm of [10] to form a hybrid procedure.

Table of Contents

What is Intersection Multiplicity (1/7)

- If p is a root of f a univariate polynomial in $\mathbb{K}[x]$, for some field $\mathbb{K}$, the multiplicity of f is the largest $m \in \mathbb{N}$ such that we can write $f = (x - p)^m g$ for some $g \in \mathbb{K}[x]$.
- When p is a solution to $f_1, \ldots, f_n \in \mathbb{K}[x_1, \ldots, x_n]$, a similar notion can be defined in this more general setting, called intersection multiplicity.
- In a sense, intersection multiplicity is the number of times a system of polynomial equations passes through a point, accounting for tangency.

What is Intersection Multiplicity (2/7)

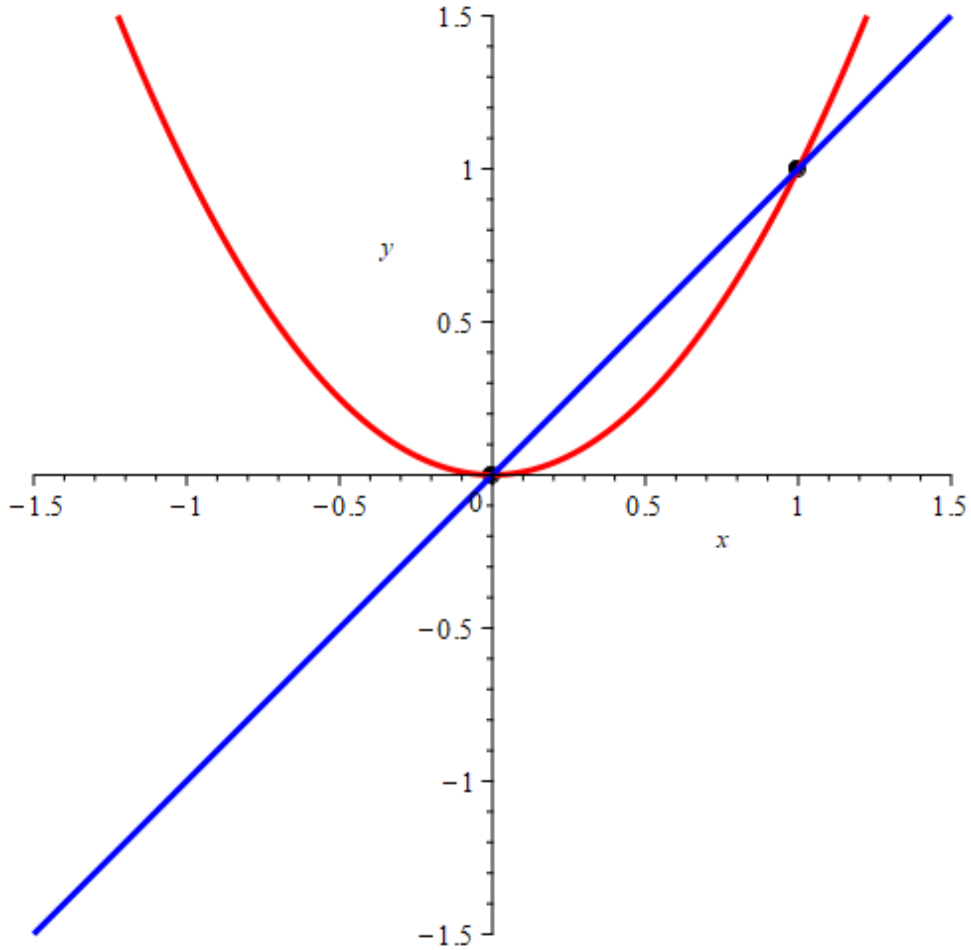

Figure: $\mathrm{Im}\left((0,0);\, y-x, y-x^2\right) = 1$
$\mathrm{Im}\left((1,1);\, y-x, y-x^2\right) = 1$

What is Intersection Multiplicity (3/7)

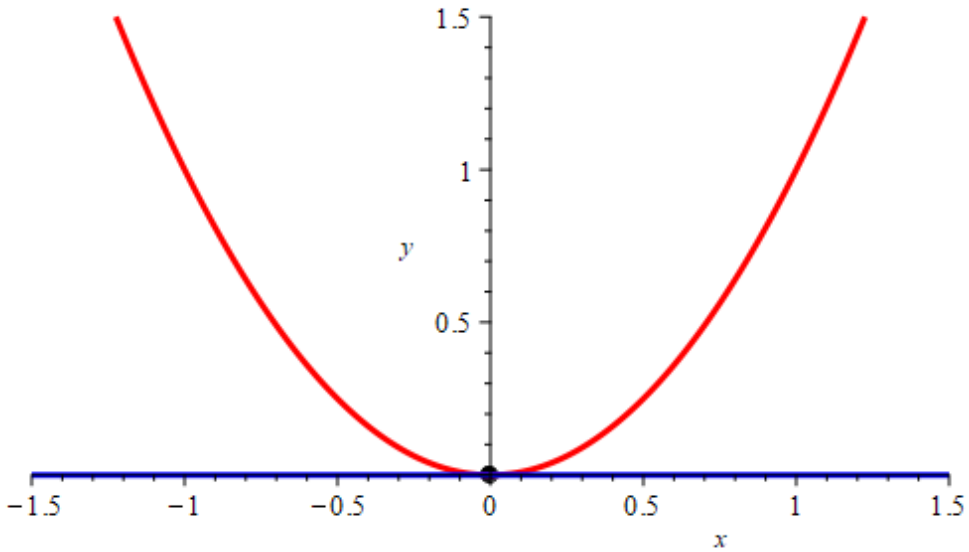

Figure: $\mathrm{Im}\left((0,0);\ y, y-x^2\right) = 2$

What is Intersection Multiplicity (4/7)

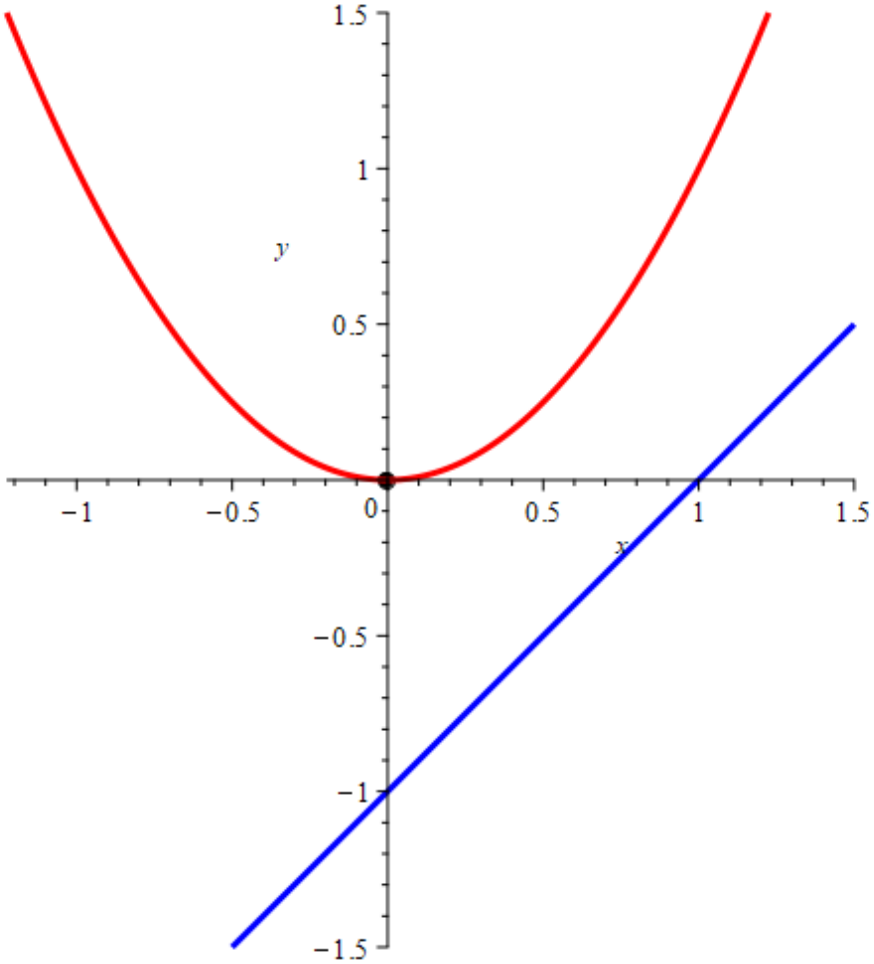

Figure: $\mathrm{Im}\left((0,0);\, y-x+1, y-x^2\right)=0$

What is Intersection Multiplicity (5/7)

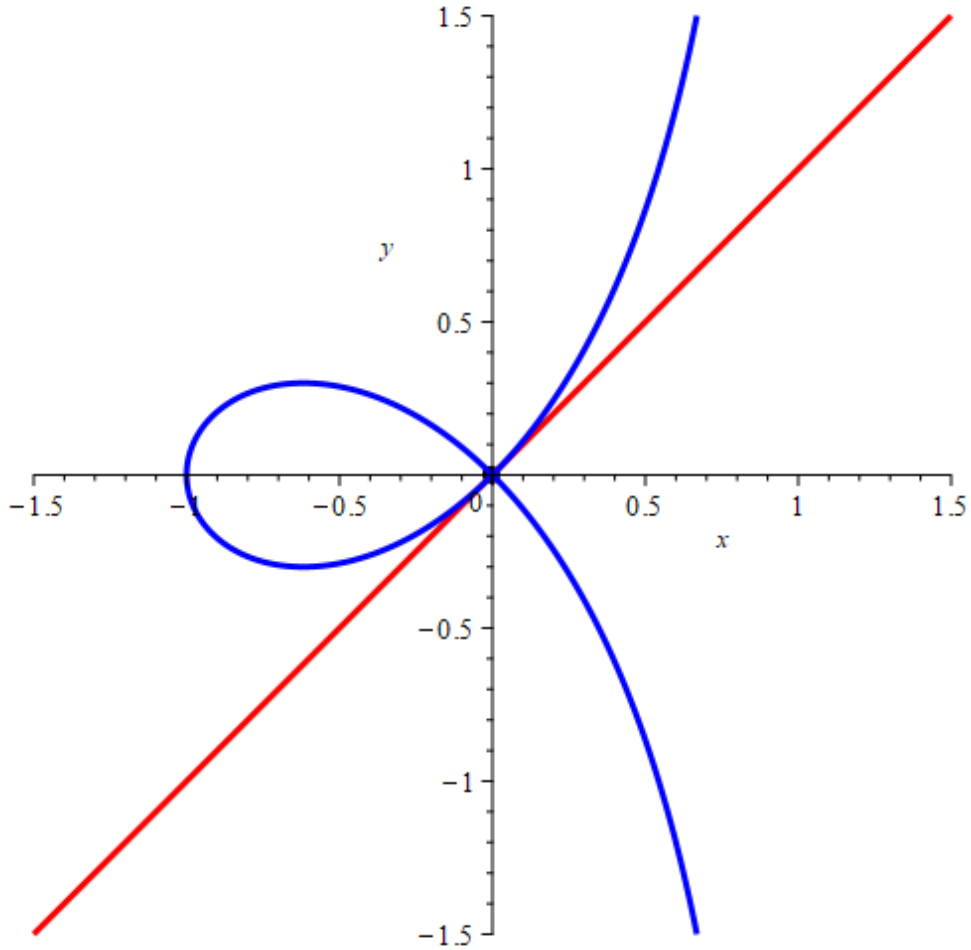

Figure: $\mathrm{Im}\left((0,0);\, x^3 + xy^2 + x^2 - y^2, y - x\right) = 3$

What is Intersection Multiplicity (6/7)

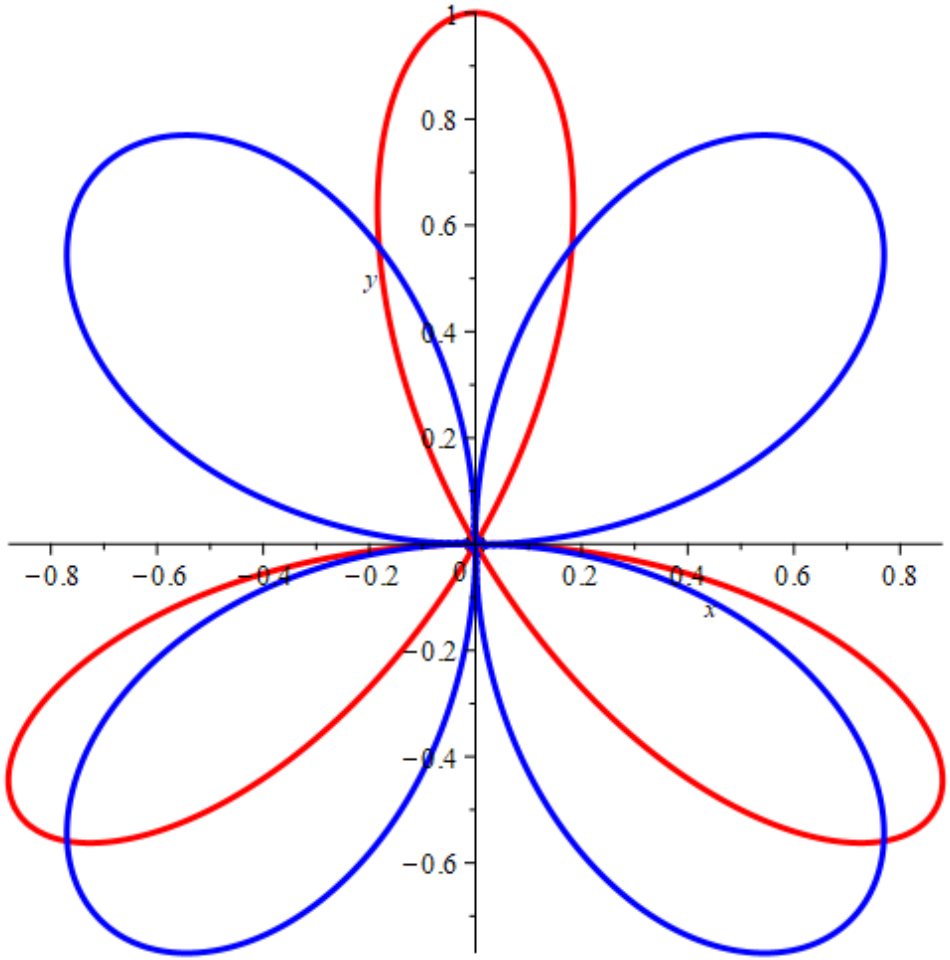

Figure: $\mathrm{Im}\left((0,0);\ (x^2+y^2)^2+3x^2y-y^3, (x^2+y^2)^3-4x^2y^2\right)=14$

What is Intersection Multiplicity (7/7)

- Let $\mathbb{K}$ be an algebraically closed field and let $\mathbb{A}^n$ denote affine n space over $\mathbb{K}$.
- We will assume all algebraic sets (varieties) are irreducible.

Definition (Local Ring)

Take $p \in \mathbb{A}^n$, we define the local ring at p as

$$\mathcal{O}_{\mathbb{A}^n,p} := \left\{ \frac{f}{g} \mid f, g \in \mathbb{K}[x_1, \ldots, x_n] \text{ where } g(p) \neq 0 \right\}.$$

Definition (Intersection Multiplicity)

Let $f_1, \ldots, f_n \in \mathbb{K}[x_1, \ldots, x_n]$. We define the intersection multiplicity of $f_1, \ldots, f_n$ at p as the dimension of the local ring at p modulo the ideal generated by $f_1, \ldots, f_n$ in the local ring at p, as a vector space over $\mathbb{K}$. That is,

$$\mathrm{Im}(p;\, f_1, \ldots, f_n) := \dim_{\mathbb{K}}(\mathcal{O}_{\mathbb{A}^n,p} / \langle f_1, \ldots, f_n \rangle).$$

Algorithms for Computing Intersection Multiplicities (1/3)

- In Algebraic Curves [5], William Fulton proved an algorithm for computing the intersection multiplicity of two bivariate, planar curves.
- Fulton's approach uses seven properties to elegantly rewrite the input system until the intersection multiplicity can either be computed or expressed as the sum of smaller intersection multiplicities.

Algorithms for Computing Intersection Multiplicities (1/2)

- The first algorithmic solution in the general setting was proposed by Mora and is described in [7].
- Mora's solution is implemented in SINGULAR's `iMult`[6] command and makes use of standard bases (Gröbner bases) to compute intersection multiplicities.
- The `iMult` command is limited to the origin, and hence, cannot compute intersection multiplicities at points other than the origin.
- Additionally, standard bases are difficult to compute in practice. The lengthy nature of standard basis computations therefore limits `iMult` to only those examples for which a standard basis can be obtained.

Algorithms for Computing Intersection Multiplicities (2/2)

- A standard basis-free intersection multiplicity algorithm was investigated in CASC 2012 and 2015 [8, 1, 10], which applies an algorithmic criterion based on tangent cones to reduce computations to the bivariate case, where Fulton's algorithm can then be applied.
- The `IntersectionMultiplicity` command, introduced in MAPLE 2020, implemented this algorithm.
- Unfortunately, the criterion discovered in [8, 1, 10] does not always apply, and hence, the `IntersectionMultiplicity` command may fail.
- We will refer to this algorithm as the algorithm of Vrbik et al.

Our Approach

- Our approach to computing intersection multiplicities without the use of standard bases, was to extend Fulton's algorithm to the n-variate case, rather then reducing to the bivariate case.
- In doing so, we successfully developed a partial algorithm which generalizes the techniques used in Fulton's algorithm to the n-variate setting.
- Our generalization is not complete as Fulton's algorithm relies on a property which is not generically true beyond the bivariate case.

Fulton's Properties

Theorem (Fulton's Properties)

Let $p = (p_1, p_2) \in \mathbb{A}^2(\mathbb{K})$ *and* $f, g \in \mathbb{K}[x, y]$.

(2-1) $\mathrm{Im}(p;\, f, g)$ *is a non-negative integer when* $\mathbf{V}(f)$ *and* $\mathbf{V}(g)$ *have no common component at* p. *When* $\mathbf{V}(f)$ *and* $\mathbf{V}(g)$ *do have a component in common at* p, $\mathrm{Im}(p;\, f, g) = \infty$.

(2-2) $\mathrm{Im}(p;\, f, g) = 0$ *if and only if* $p \notin \mathbf{V}(f, g)$.

(2-3) $\mathrm{Im}(p;\, f, g)$ *is invariant under affine changes of coordinates on* $\mathbb{A}^2$.

(2-4) $\mathrm{Im}(p;\, f, g) = \mathrm{Im}(p;\, g, f)$.

(2-5) $\mathrm{Im}(p;\, (x - p_1)^{m_1}, (y - p_2)^{m_2}) = m_1 m_2$ *for* $m_1, m_2 \in \mathbb{N}$.

(2-6) $\mathrm{Im}(p;\, f, gh) = \mathrm{Im}(p;\, f, g) + \mathrm{Im}(p;\, f, h)$ *for any* $h \in \mathbb{K}[x, y]$ *such that* $\mathrm{Im}(p;\, f, gh) \in \mathbb{N}$.

(2-7) $\mathrm{Im}(p;\, f, g) = \mathrm{Im}(p;\, f, g + hf)$ *for any* $h \in \mathbb{K}[x, y]$.

Fulton's Algorithm

Algorithm 1: Fulton's algorithm

1 **Function** $\mathrm{im}_2(f, g)$

Input: Let: $x \succ y$

1. $f, g \in \mathbb{K}[x, y]$ such that $\gcd(f, g)(0, 0) \neq 0$.

Output: $\mathrm{Im}((0, 0); f, g)$

2 **if** $f(0, 0) \neq 0$ **or** $g(0, 0) \neq 0$ **then**

3 **return** 0

4 $r \leftarrow \deg_x(f(x, 0))$

5 $s \leftarrow \deg_x(g(x, 0))$

6 **if** $r > s$ **then**

7 **return** $\mathrm{im}_2(g, f)$

8 **if** $r < 0$ **then** /* $y \mid f$ */

9 write $g(x, 0) = x^m(a_m + a_{m+1}x + \ldots)$

/* $\mathrm{im}_2(f, g) = \mathrm{im}_2(\mathrm{quo}(f, y; y), g) + \mathrm{im}_2(y, g)$ */

10 **return** $\mathrm{im}_2(\mathrm{quo}(f, y; y), g) + m$

11 **else**

12 $g' = \mathrm{lc}(f(x, 0)) \cdot g - (x)^{s-r}\mathrm{lc}(g(x, 0)) \cdot f$

13 **return** $\mathrm{im}_2(f, g')$

Table of Contents

Regular Sequences

- An assumption of Fulton's algorithm states that the input polynomials must not have a common factor which vanishes at p, where $p \in \mathbb{A}^2$ is the point which we wish to compute the intersection multiplicity over.
- To generalize this condition we must introduce the notion of a regular sequence.
- Roughly, this means given $f_1, \ldots, f_n$ as input, no f_i is a unit, zero, or a zero-divisor modulo the ideal generated by any subset of the remaining input polynomials, in the local ring at p.

Theorem

Let I be the ideal generated by $f_1, \ldots, f_n \in \mathbb{K}[x_1, \ldots, x_n]$ and define $\mathbf{V} = \mathbf{V}(I)$. Suppose $\mathbf{V}$ is non-empty and irreducible, then for any $p \in \mathbf{V}$, $\dim(\mathbf{V}) = 0$ if and only if $f_1, \ldots, f_n$ is a regular sequence in $\mathcal{O}_{\mathbb{A}^n, p}$.

Generalizing Fulton's Properties

Theorem (Fulton's Properties)

Let $p = (p_1, \ldots, p_n) \in \mathbb{A}^n$ *and* $f_1, \ldots, f_n \in \mathbb{K}[x_1, \ldots, x_n]$.

(n-1) $\mathrm{Im}(p;\, f_1, \ldots, f_n)$ *is a non-negative integer if and only if* $\mathbf{V}(f_1, \ldots, f_n)$ *is zero-dimensional.*

(n-2) $\mathrm{Im}(p;\, f_1, \ldots, f_n) = 0$ *if and only if* $p \notin \mathbf{V}(f_1, \ldots, f_n)$.

(n-3) $\mathrm{Im}(p;\, f_1, \ldots, f_n)$ *is invariant under affine changes of coordinates on* $\mathbb{A}^n$.

(n-4) $\mathrm{Im}(p;\, f_1, \ldots, f_n) = \mathrm{Im}\left(p;\, f_{\sigma(1)}, \ldots, f_{\sigma(n)}\right)$.

(n-5) $\mathrm{Im}(p;\, (x_1 - p_1)^{m_1}, \ldots, (x_n - p_n)^{m_n}) = m_1 \cdot \ldots \cdot m_n$ *for* $m_1, \ldots, m_n \in \mathbb{N}$.

(n-6) $\mathrm{Im}(p;\, f_1, \ldots, gh) = \mathrm{Im}(p;\, f_1, \ldots, g) + \mathrm{Im}(p;\, f_1, \ldots, h)$ *for any* $g, h \in \mathbb{K}[x_1, \ldots, x_n]$ *such that* $f_1, \ldots, gh$ *is a regular sequence in* $\mathcal{O}_{\mathbb{A}^n, p}$.

(n-7) $\mathrm{Im}(p;\, f_1, \ldots, f_n) = \mathrm{Im}(p;\, f_1, \ldots, f_n + g)$ *for any* $g \in \langle f_1, \ldots, f_{n-1} \rangle$.

Modular Degrees

- In Fulton's algorithm, we considered $r = \deg_x(f(x,0)), s = \deg_x(g(x,0))$ for $f, g \in \mathbb{K}[x,y]$.
- Using r, s we then determined an appropriate rewrite rule to apply to f, g.
- The following definition generalizes this notion to the n-variate case.

Definition (Modular Degree)

Take $p \in \mathbb{A}^n$, $v \in \{x_1, \ldots, x_n\}$, and $f \in \mathbb{K}[x_1, \ldots, x_n]$ where $x_1 \succ \ldots \succ x_n$. We define the modular degree of f at p with respect to v as

$$\deg_v(f \mod \langle V_{<v,p} \rangle),$$

where $V_{<v,p} = \{x_i - p_i | x_i \prec v\}$. If $V_{<v,p} = \emptyset$ then the modular degree of f at p with respect to v is simply the degree of f with respect to v.

- Often we will assume p is the origin, in which case we write $\mathrm{moddeg}(f, v)$ to denote the modular degree of f at v.

Splitting

- The following lemma uses modular degrees to generalize the conditions of the splitting (yellow) case in Fulton's algorithm.

Lemma

Let $f_1, \ldots, f_n \in \mathbb{K}[x_1, \ldots, x_n]$ *forming a regular sequence in* $\mathcal{O}_{\mathbb{A}^n, p}$ *where* p *is the origin. Let* $V = \{x_1, \ldots, x_n\}$ *and let* $V_{>v} = \{x_i \in V \mid x_i > v\}$. *Define* $J : \{1, \ldots, n\} \to \{1, \ldots, n\}$ *such that* $J(i) = n - i + 1$.

Assume $\operatorname{moddeg}(f_i, v) < 0$ *holds for all* $i = 1, \ldots, n-1$ *and all* $v \in V_{>x_{J(i)}}$. *Then, we have* $x_{J(i)} \mid f_i(x_1, \ldots, x_{J(i)}, 0, \ldots, 0)$. *Moreover, if we define* $q_i = quo(f_i(x_1, \ldots, x_{J(i)}, 0, \ldots, 0), x_{J(i)}; x_{J(i)})$ *then,*

$$\begin{aligned} \operatorname{Im}(p;\, f_1, \ldots, f_n) &= \operatorname{Im}(p;\, q_1, f_2, \ldots, f_n) + \operatorname{Im}(p;\, x_n, q_2, \ldots, f_n) \\ &+ \ldots + \operatorname{Im}\left(p;\, x_n, \ldots, x_{J(i)+1}, q_i, f_{i+1}, \ldots, f_n\right) + \ldots \\ &+ \operatorname{Im}(p;\, x_n, x_{n-1}, \ldots, q_{n-1}, f_n) + m_n \end{aligned}$$

where $m_n = max(m \in \mathbb{Z}^+ \mid f_n(x_1, 0, \ldots, 0) \equiv 0 \mod \langle x_1^m \rangle)$.

Splitting Continued

- Simply put, the above lemma states when the matrix of modular degrees has a triangular shape we may split the intersection multiplicity into smaller computations.

Example

Let $f_1, f_2, f_3 \in \mathbb{K}[x_1, x_2, x_3]$, let $x_1 \succ x_2 \succ x_3$ and let $p \in \mathbb{A}^3$ be the origin. Take $f_1 = x_3(x_2 + 1), f_2 = x_2^3 + x_3, f_3 = x_3^2 + x_2^3 + x_1^4(x_1 + 1)$. Write R the matrix of modular degrees such that $R_{i,j} = \operatorname{moddeg}(f_i, x_j)$

$$R = \begin{bmatrix} -\infty & -\infty & 1 \\ -\infty & 3 & 1 \\ 4 & 3 & 2 \end{bmatrix},$$

$$\begin{aligned} & \operatorname{Im}(p;\, f_1, f_2, f_3) = \operatorname{Im}(p;\, x_2 + 1, f_2, f_3) + \operatorname{Im}(p;\, x_3, f_2, f_3) \\ = \;& \operatorname{Im}(p;\, x_2 + 1, f_2, f_3) + \operatorname{Im}\left(p;\, x_3, x_2^2, f_3\right) + \operatorname{Im}(p;\, x_3, x_2, f_3) \\ = \;& 0 + 8 + 4 \end{aligned}$$

Why The Generalization is Not Complete

- Assume p is the origin.
- In the bivariate case for polynomials $f, g \in \mathbb{K}[x, y]$, one step of the algorithm may replace g with $g' := \mathrm{lc}(f(x,0))g - x^d \mathrm{lc}(g(x,0))f$ for some $d \in \mathbb{N}$. This preserves intersection multiplicity since $\mathrm{lc}(f(x,0)) \in \mathbb{K}$ and hence $\langle f, g \rangle = \langle f, g' \rangle$, i.e. property (2-7) applies.
- When we generalize this, say with polynomials $f_1, \ldots, f_n \in \mathbb{K}[x_1, \ldots, x_n]$, we replace some f_i with

  $$f_i' := \mathrm{lc}(f_m(x_1, \ldots, x_k, 0, \ldots, 0); x_k) f_i - x_k^d \mathrm{lc}(f_i(x_1, \ldots, x_k, 0, \ldots, 0); x_k) f_m$$

  for some $i, m, k, d \in \mathbb{N}, i \neq m$.
- Unlike the bivariate case, $\mathrm{lc}(f_m(x_1, \ldots, x_k, 0, \ldots, 0); x_k)$ is not always invertible in $\mathcal{O}_{\mathbb{A}^n, p}$, hence property ($n$-7) does not always apply.
- Hence, it is not generically true that $\langle f_1, \ldots, f_i, \ldots, f_n \rangle = \langle f_1, \ldots, f_i', \ldots, f_n \rangle$. That is, substituting f_i' for f_i does not necessarily preserve intersection multiplicity.

Generalizing Fulton's Algorithm (1/3)

Algorithm 2: The Generalization of Fulton's Algorithm

Function $\mathrm{im}_n(p; f_1, \ldots, f_n)$

Input: Let: $x_1 \succ \ldots \succ x_n$.

- $p \in \mathbb{A}^n$ the origin.
- $f_1, \ldots, f_n \in \mathbb{K}[x_1, \ldots, x_n]$ such that $f_1, \ldots, f_n$ form a regular sequence in $\mathcal{O}_{\mathbb{A}^n,p}$ or at least one such f_i is a unit in $\mathcal{O}_{\mathbb{A}^n,p}$.

Output: $\mathrm{Im}(p; f_1, \ldots, f_n)$ or Fail

if $f_i(p) \neq 0$ *for any* $i = 1, \ldots, n$ **then**

  return 0

if $n = 1$ **then** /* Compute multiplicity */

  return $\max(m \in \mathbb{Z}^+ \mid f_n \equiv 0 \mod \langle x_1^m \rangle)$

for $i = 1, \ldots, n$ **do**

  for $j = 1, \ldots, n-1$ **do**

    $r_j^{(i)} \leftarrow \mathrm{moddeg}(f_i, x_j)$

- In this first section of our extension of Fulton's algorithm, we check if any of the polynomials do not vanish at p. That is, we check the base case of the algorithm.
- We also compute the $n \times n-1$ matrix of modular degrees, analogous to computing r, s in the bivariate algorithm.

Example

Let $f_1, f_2, f_3 \in \mathbb{K}[x, y, z]$ be given by $f_1 = x^2, f_2 = (x+1)y + x^3, f_3 = y^2 + z + x^3$. The matrix r computed in the first section of our generalization is:

$$r = \begin{bmatrix} 2 & 0 \\ 3 & 1 \\ 3 & 2 \end{bmatrix},$$

where the i-th row corresponds to the polynomial f_i and the j-th column corresponds to the variable x_j. Hence, (i, j)-th entry is the modular degree of f_i with respect to x_j.

Generalizing Fulton's Algorithm (2/3)

9
10 **for** $j = 1, \ldots, n-1$ **do**
11 Reorder $f_1, \ldots, f_{n-j+1}$ so that $r_j^{(1)} \leq \ldots \leq r_j^{(n-j+1)}$
12 $m \leftarrow min(i \mid r_j^{(i)} > 0)$ or $m \leftarrow \infty$ if no such i exists
13 **if** $m \leq (n-j)$ **then**
14 **for** $i = m+1, \ldots, n-j+1$ **do**
15 $d \leftarrow r_j^{(i)} - r_j^{(m)}$
16 $L_m \leftarrow \mathrm{lc}(f_m(x_1, \ldots, x_j, 0, \ldots, 0); x_j)$
17 $L_i \leftarrow \mathrm{lc}(f_i(x_1, \ldots, x_j, 0, \ldots, 0); x_j)$
18 **if** $L_m(p) \neq 0$ **then**
19 $f_i' \leftarrow L_m f_i - x_j^d L_i f_m$
20 **else if** $L_m \mid L_i$ **then**
21 $f_i' \leftarrow f_i - x_j^d \frac{L_i}{L_m} f_m$
22 **else**
23 **return** Fail
24 **return** $\mathrm{im_n}(p; f_1, \ldots, f_m, f_{m+1}', \ldots, f_{n-j+1}', \ldots, f_n)$

- In the second section, we seek to transform the matrix of modular degrees into a triangular shape.
- Namely, we seek to transform the matrix of modular degrees so that all entries above the anti-diagonal are $-\infty$.
- This is done through reordering polynomials by modular degree and applying (n-7) to rewriting polynomials in a way that decreases their modular degree with respect to x_j.

Example

Let $f_1, f_2, f_3 \in \mathbb{K}[x, y, z]$ be as above, given by $f_1 = x^2, f_2 = (x+1)y + x^3, f_3 = y^2 + z + x^3$. Write

$$f_2' := f_2 - xf_1 = (x+1)y + x^3 - x^3 = (x+1)y,$$

and

$$f_3' := f_3 - xf_1 = y^2 + z + x^3 - x^3 = y^2 + z.$$

Example (continued)

Redefine $f_2 := f_2'$ and $f_3 := f_3'$. Hence, we consider $f_1 = x^2, f_2 = (x+1)y, f_3 = y^2 + z$. The matrix r computed in the first section is now:

$$r = \begin{bmatrix} 2 & 0 \\ -\infty & 1 \\ -\infty & 2 \end{bmatrix},$$

and after reordering f_1, f_2, f_3 by modular degree we have $f_1 = (x+1)y, f_2 = y^2 + z, f_3 = x^2$, with matrix of modular degrees:

$$r = \begin{bmatrix} -\infty & 1 \\ -\infty & 2 \\ 2 & 0 \end{bmatrix}.$$

- Now we apply the same procedure to reduce the modular degree with respect to y.

Example (continued)

Consider $f_1 = (x+1)y, f_2 = y^2 + z, f_3 = x^2$. Write

$$f_2' := (x+1)f_2 - yf_1 = (x+1)y^2 + (x+1)z - (x+1)y^2 = (x+1)z.$$

Redefining $f_2 := f_2'$ and reordering by modular degree in y, we have $f_1 = (x+1)z, f_2 = (x+1)y, f_3 = x^2$, and the matrix of modular degrees is now:

$$r = \begin{bmatrix} -\infty & -\infty \\ -\infty & 1 \\ 2 & 0 \end{bmatrix}.$$

- Notice (n-7) applies when computing f_2' above since $x+1$ is invertible in $\mathcal{O}_{\mathbb{A}^n,p}$.

Generalizing Fulton's Algorithm (3/3)

25

26

27 **for** $i = 1, \ldots, n-1$ **do**

28 $\quad q_i \leftarrow \mathrm{quo}(f_i(x_1, \ldots, x_{n-i+1}, 0, \ldots, 0), x_{n-i+1}; x_{n-i+1})$

29 **return**

30 $\mathrm{im_n}(p; q_1, f_2, \ldots, f_n)$

31 $+ \mathrm{im}_{n-1}(p; q_2(x_1, \ldots, x_{n-1}, 0), \ldots, f_n(x_1, \ldots, x_{n-1}, 0))$

32 $+$

33 $\vdots$

34 $+\mathrm{im}_2(p; q_{n-1}(x_1, x_2, 0, \ldots, 0), f_n(x_1, x_2, 0, \ldots, 0))$

35 $+\mathrm{im}_1(p; f_n(x_1, 0, \ldots, 0))$

Example (continued)

Applying the splitting lemma on $f_1 = (x+1)z, f_2 = (x+1)y, f_3 = x^2$ gives:

$$\begin{aligned}
&\mathrm{Im}(p;\, f_1, f_2, f_3) \\
&= \mathrm{Im}(p;\, x+1, f_2, f_3) + \mathrm{Im}(p;\, z, f_2, f_3) \\
&= \mathrm{Im}(p;\, x+1, f_2, f_3) + \mathrm{Im}(p;\, z, x+1, f_3) + \mathrm{Im}(p;\, z, y, f_3) \\
&= 0 + 0 + 2.
\end{aligned}$$

Table of Contents

Fulton's Algorithm and its Generalization, with Non-Rational Points

- Fulton's algorithm, and the generalization of Fulton's algorithm, assume that the point p is the origin. If p has rational coordinates then one can easily reduce to the case where p is the origin.
- When p does not have rational coordinates, it is often not practical to represent p by its coordinates. For example, consider any point p which is a solution to $x^{10000} = 2, y^2 = x + 1$.
- To avoid this, it is natural to encode p as a solution to a system of polynomial equations, rather than as a list of algebraic numbers.
- By encoding p as a solution to a zero-dimensional regular chain, we can adapt Fulton's algorithm and its generalization to work with any point p of the zero set $\mathbf{V}(F)$ of F.

Extending the Generalization of Fulton's Algorithm Using Regular Chains

- In our implementation of the generalization of Fulton's algorithm, we extended Algorithm 2 to handle a zero-dimensional regular chain as input, rather than a point.
- A zero-dimensional regular chain is a triangular system of equations with algorithmic properties which encode a finite set of points.
- Since zero-dimensional regular chains may encode many points with different intersection multiplicities, the new version of the algorithm may return multiple intersection multiplicities, and the points they correspond to, encoded in zero-dimensional regular chains.

Example

$$with(RegularChains):$$
$$with(AlgebraicGeometryTools):$$
$$with(ChainTools):$$

$$R := PolynomialRing([x, y, z]):$$
$$rc := Chain\left(\left[z^6 + 2, y^2 - z, x\cdot(x-1)\right], R\right):$$
$$F := \left[x(x-1), xy^4 - 2xy^2z + xz^2 + \left(z^6+2\right)^3, z^5y^2 + 2\right]:$$

$$dec := IntersectionMultiplicity(rc, F, R);$$
$$Display(dec, R);$$

$$dec := [[2, regular\_chain], [3, regular\_chain]]$$

$$\left[\left[2, \left\{\begin{array}{l} x-1=0 \\ y^2-z=0 \\ z^6+2=0 \end{array}\right\}\right], \left[3, \left\{\begin{array}{l} x=0 \\ y^2-z=0 \\ z^6+2=0 \end{array}\right\}\right]\right]$$

Figure: The generalization of Fulton's algorithm applied to F at a regular chain encoding 24 points.

Table of Contents

Modifications to `IntersectionMultiplicity` (1/2)

- We have extended MAPLE's `IntersectionMultiplicity` command to a hybrid algorithm which first calls the generalization of Fulton's algorithm, and upon detecting failure, calls the algorithm of Vrbik et al.
- This implementation supports both, points passed as a list of rational coordinates, and points encoded by a zero-dimensional regular chain.
- Both the a generalization of Fulton's algorithm and the algorithm of Vrbik et al. can be accessed individually using the `method` keyword.
- A special calling sequence was also included to optimize for the special case of triangular regular sequences.

Modifications to `IntersectionMultiplicity` (2/2)

- Our implementation improves on the pivot selection process of Algorithm 2, increasing the algorithm's chance of success by trying all polynomials with minimal modular degree as a pivot.
- `IntersectionMultiplicity` also supports changes in the variable ordering, manually, by modifying the parameters passed to the command, or through the `maxshift` option, which applies a fixed number of circular left shifts to the variable ordering.

Modifications to `TriangularizeWithMultiplicity`

- In order to compute a non-zero intersection multiplicity, one first needs a solution to a given polynomial system. In practice one may not always know the solutions of a polynomial system.
- `TriangularizeWithMultiplicity` addresses this problem by first solving the system using the `Triangularize` command of the `RegularChains` library, and then, calls the `IntersectionMultiplicity` command on each of the solutions returned by `Triangularize`.
- We hence, modify `TriangularizeWithMultiplicity` to support the new version of the `IntersectionMultiplicity` command, and include it in our experiments.

TriangularizeWithMultiplicity Example

$$R := PolynomialRing([x, z, y]) :$$
$$F := \left[6x^4z^2 - 3x^2y^2z^2 - x^2z^2 + 28x^2z - 3y^4z^3 + 2y^2z^2 + 7y^2z + z^2 - 11z + 10,\ x + x^3z + xy^2z - xz,\ 10y - 2x^2yz - y^3z - yz \right] :$$
$$dec := TriangularizeWithMultiplicity(F, R);$$
$$Display(dec, R);$$

$$dec := [[1, regular\_chain], [1, regular\_chain], [1, regular\_chain], [3, regular\_chain], [3, regular\_chain]]$$

$$\left[\left[1, \left\{ \begin{array}{c} 12x^2 + 11y^2 - 9 = 0 \\ 324z - 113y^4 - 270y^2 - 1377 = 0 \\ 113y^6 - 69y^4 + 567y^2 - 243 = 0 \end{array} \right\}\right], \left[1, \left\{ \begin{array}{c} x = 0 \\ 15z + 4y^2 - 141 = 0 \\ 4y^4 - 137y^2 + 9 = 0 \end{array} \right\}\right], \left[1, \left\{ \begin{array}{c} 2x^2 - 3 = 0 \\ z + 2 = 0 \\ y = 0 \end{array} \right\}\right], \left[3, \left\{ \begin{array}{c} x = 0 \\ z - 1 = 0 \\ y = 0 \end{array} \right\}\right], \left[3, \left\{ \begin{array}{c} x = 0 \\ z - 10 = 0 \\ y = 0 \end{array} \right\}\right]\right]$$

Figure: `TriangularizeWithMultiplicity` applied to the Ojika4 system.

Ojika4

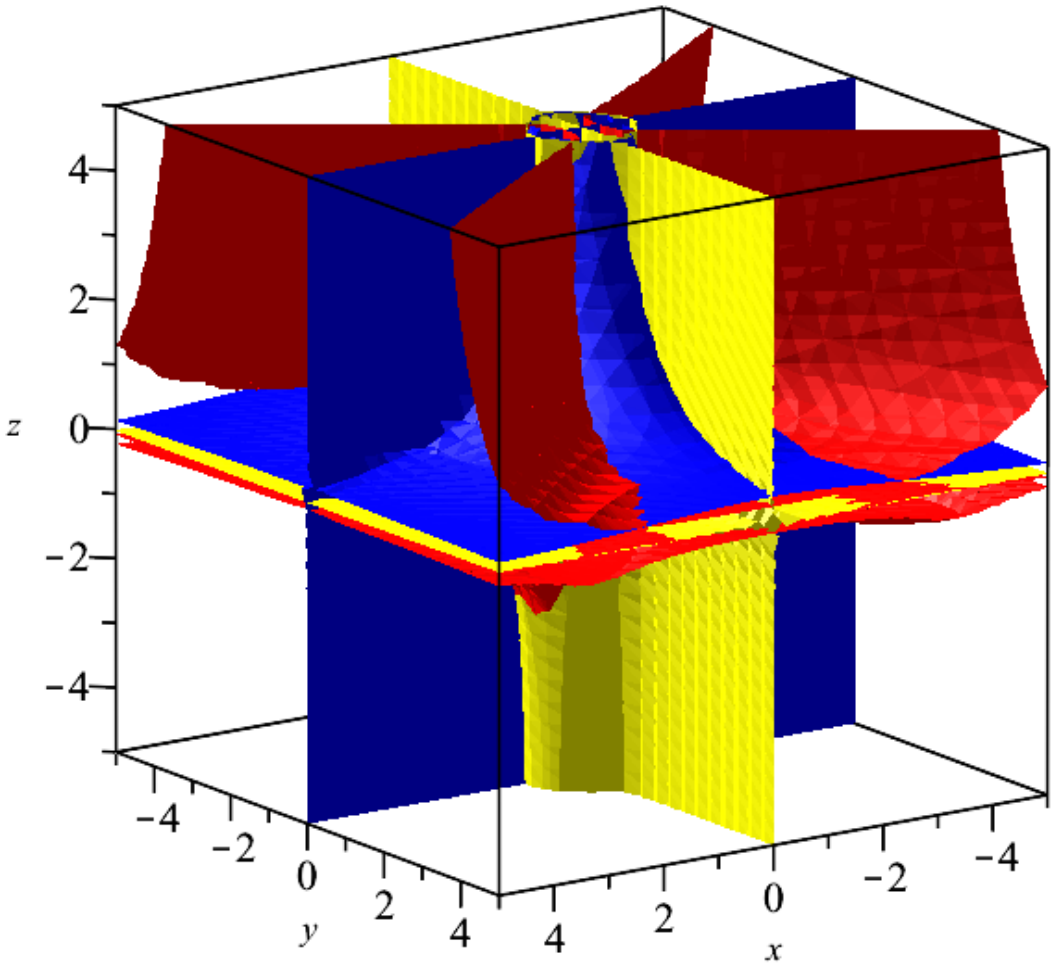

Figure: The Ojika4 system.

Table of Contents

Legend I

- n denotes the size of the square system, that is the number of variables and number of polynomials in the system.
- The point column represents the point encoded by the zero-dimensional regular chain passed as input.
- The ordering column denotes the variable ordering given to the regular chain. This variable ordering is the same ordering used by the generalization of Fulton's algorithm in our implementation.
- Columns under the heading Fulton denote results obtained from calling the generalization of Fulton's algorithm and columns under the heading Vrbik denote results obtained from calling the algorithm of Vrbik et al.
- The Im column denotes the intersection multiplicity computed using the given algorithm and CPU time denotes the CPU time elapsed during the respective computation.
- We use NR to denote no result.

Experiments (1/6)

Table: `IntersectionMultiplicity` Using the Author's Tests

	System Specifications			Fulton		Vrbik	
System	n	Ordering	Point	Im	CPU Time	Im	CPU Time
test 1	3	$x_1 \succ \ldots \succ x_3$	origin	2	15.00ms	2	329.00ms
test 2	5	$x_1 \succ \ldots \succ x_5$	origin	2	31.00ms	2	2.95s
test 3	7	$x_1 \succ \ldots \succ x_7$	origin	2	94.00ms	2	8.84s
test 4	9	$x_1 \succ \ldots \succ x_9$	origin	2	188.00ms	2	20.69s
test 5	11	$x_1 \succ \ldots \succ x_{11}$	origin	2	359.00ms	2	40.39s
test 6	13	$x_1 \succ \ldots \succ x_{13}$	origin	2	610.00ms	2	71.58s
test 7	15	$x_1 \succ \ldots \succ x_{15}$	origin	2	1.17s	2	118.52s
test 8	25	$x_1 \succ \ldots \succ x_{25}$	origin	2	7.81s	2	17.32m
test 9	3	$x \succ y \succ z$	origin	5	47.00ms	NR	NR
test 10	3	$x \succ y \succ z$	origin	24	109.00ms	ERROR	16.00ms
test 11	3	$x \succ y \succ z$	origin	45	93.00ms	ERROR	15.00ms
test 12	6	$x_1 \succ \ldots \succ x_6$	origin	2	125.00ms	2	18.09s

Experiments (2/6)

Table: `IntersectionMultiplicity` Using Examples from the Literature

	System Specifications			Fulton		Vrbik	
System	n	Ordering	Point	Im	CPU Time	Im	CPU Time
cbms1	3	$x \succ y \succ z$	$(0,0,0)$	FAIL	32.00ms	ERROR	16.00ms
cbms2	3	$x \succ y \succ z$	$(0,0,0)$	FAIL	31.00ms	ERROR	31.00ms
mth191	3	$x \succ y \succ z$	$(0,1,0)$	4	31.00ms	ERROR	1.47s
decker1	2	$x \succ y$	$(0,0)$	3	15.00ms	3	171.00ms
decker2	2	$x \succ y$	$(0,0)$	4	14.00ms	4	187.00ms
Ojika2	3	$x \succ y \succ z$	$(0,0,1)$	2	63.00ms	2	1.53s
Ojika2	3	$x \succ y \succ z$	$(1,0,0)$	2	62.00ms	2	1.48s
Ojika3	3	$x \succ y \succ z$	$(0,0,1)$	4	31.00ms	4	1.62s
Ojika3	3	$x \succ y \succ z$	$(-\frac{5}{2},\frac{5}{2},1)$	2	31.00ms	2	1.02s
Ojika4	3	$x \succ y \succ z$	$(0,0,1)$	FAIL	16.00ms	ERROR	657.00ms
Ojika4	3	$x \succ y \succ z$	$(0,0,10)$	FAIL	16.00ms	3	2.00s
Ojika4	3	$x \succ z \succ y$	$(0,1,0)$	3	46.00ms	ERROR	2.50s
Ojika4	3	$x \succ z \succ y$	$(0,10,0)$	3	63.00ms	3	4.02s
Caprasse	4	$x_1 \succ \ldots \succ x_4$	$(2,-i\sqrt{3},2,i\sqrt{3})$	FAIL	94.00ms	NR	>2000s
KSS	5	$x_1 \succ \ldots \succ x_5$	$(1,1,1,1,1)$	FAIL	43.00ms	ERROR	56.94s
DZ1	4	$x_1 \succ \ldots \succ x_4$	$(0,0,0,0)$	FAIL	31.00ms	ERROR	16.00ms
DZ2	3	$x \succ z \succ y$	$(0,0,-1)$	16	78.00ms	ERROR	16.00ms
Solotarev	4	$x_1 \succ \ldots \succ x_4$	$(\frac{5}{3},-1,5,-\frac{47}{27})$	2	110.00ms	2	1.36s
Solotarev	4	$x_1 \succ \ldots \succ x_4$	$(-1,-1,5,3)$	2	93.00ms	2	1.36s

Legend II

- Since $\mathrm{TriangularizeWithMultiplicity}$ may return many regular chains, we define Success Ratio to be the ratio of intersection multiplicities successfully computed to the number of regular chains returned, using the generalization of Fulton's algorithm and the alforithm of Vrbik et al.
- The implementation of the algorithm of Vrbik et al. throws an error is thrown if it is unable to compute all intersection multiplicities, hence the column Success Ratio will always contain a full fraction, an error, or NR when this algorithm is used.

Experiments (3/6)

Table: `TriangularizeWithMultiplicity` Using Examples from the Literature

	System Specifications		Fulton		Vrbik	
System	n	Ordering	Success Ratio	CPU Time	Success Ratio	CPU Time
cbms1	3	$x \succ y \succ z$	10/11	641.00ms	ERROR	218.00ms
cbms2	3	$x \succ y \succ z$	1/2	11.55s	ERROR	422.00ms
mth191	3	$x \succ y \succ z$	8/8	609.00ms	ERROR	2.45s
decker1	2	$x \succ y$	3/3	47.00ms	3/3	140.00ms
decker2	2	$x \succ y$	3/3	63.00ms	3/3	250.00ms
Ojika2	3	$x \succ y \succ z$	4/4	219.00ms	4/4	5.38s
Ojika3	3	$x \succ y \succ z$	2/2	62.00ms	2/2	5.02s
Ojika4	3	$x \succ y \succ z$	3/5	438.00ms	ERROR	906.00ms
Ojika4	3	$x \succ z \succ y$	5/5	500.00ms	ERROR	3.41s
Caprasse	4	$x_1 \succ x_2 \succ \ldots$	4/15	1.58s	NR	>2000s
Caprasse	4	$x_4 \succ x_2 \succ x_1 \succ x_3$	12/15	4.48s	ERROR	14.09s
KSS	5	$x_1 \succ x_2 \succ \ldots$	16/17	3.34s	ERROR	71.16s
DZ1	4	$x_1 \succ x_2 \succ \ldots$	NR	>2000s	ERROR	313.00ms
DZ2	3	$x \succ z \succ y$	2/2	172.00ms	ERROR	47.00ms
Solotarev	4	$x_1 \succ x_2 \succ \ldots$	4/4	453.00ms	4/4	3.47s

Experiments (4/6)

Table: `TriangularizeWithMultiplicity` Using Examples from the Literature with Multiplicity 1

	System Specifications	Fulton	
System	n	Success Ratio	CPU Time
eco5	5	4/4	609.00ms
eco6	6	5/5	2.52s
eco7	7	6/6	97.17s
trinks	6	2/2	516.00ms
Czapor Geddes 1	3	1/1	281.00ms
A Bifurcation Problem	3	3/3	2.95m
quadfor2	4	1/1	109.00ms
Lorentz	4	6/6	485.00ms
S9_1	8	2/2	1.80s
cyclic3	3	2/2	110.00ms

Legend III

- Columns under the heading `iMult` denote results obtained from calling the `iMult` command and columns under the heading `IntersectionMultiplicity` denote results obtained from calling the generalization of Fulton's algorithm.
- The ordering column denotes the variable ordering given to either the generalization of Fulton's algorithm in the case of `IntersectionMultiplicity`, or the variable ordering given to the polynomial ring in the case of `iMult`.
- The Im column denotes the intersection multiplicity computed using the given algorithm and CPU time denotes the CPU time elapsed during the respective computation.
- For tests run using either MAPLE or SINGULAR, if we experience a timeout or error, we populate all larger systems in the same family of experiments with that result. That is, all cells in the given column with the same parameter d and larger n, will be automatically populated with the notation for either a timeout or error, respectively.

Experiments (5/6)

Table: `iMult` and `IntersectionMultiplicity` on nql-*n*-*d*

	iMult				IntersectionMultiplicity			
Order	$x_1 \succ \ldots \succ x_n$		$x_1 \prec \ldots \prec x_n$		$x_1 \succ \ldots \succ x_n$		$x_1 \prec \ldots \prec x_n$	
System	Im	CPU Time	Im	CPU Time	Im	CPU Time	Im	CPU Time
nql-3-4	16	<1.00ms	16	<1.00ms	16	109.00ms	16	16.00ms
nql-4-4	32	<1.00ms	32	<1.00ms	32	625.00ms	32	16.00ms
nql-5-4	64	48.16s	64	20.00ms	64	4.55s	64	47.00ms
nql-6-4	NR	>2000s	128	50.00ms	128	37.75s	128	63.00ms
nql-7-4	NR	>2000s	256	310.00ms	256	5.51m	256	109.00ms
nql-8-4	NR	>2000s	512	3.2s	NR	>2000s	512	203.00ms
nql-9-4	NR	>2000s	1024	79.5s	NR	>2000s	1024	422.00ms
nql-10-4	NR	>2000s	NR	ERROR	NR	>2000s	2048	828.00ms
nql-3-6	54	<1.00ms	54	<1.00ms	54	469.00ms	54	31.00ms
nql-4-6	162	19.21s	162	3.71m	162	5.38s	162	62.00ms
nql-5-6	NR	>2000s	NR	>2000s	486	86.11s	486	125.00ms
nql-6-6	NR	>2000s	NR	>2000s	NR	>2000s	1458	313.00ms
nql-7-6	NR	>2000s	NR	>2000s	NR	>2000s	4374	1.66s
nql-8-6	NR	>2000s	NR	>2000s	NR	>2000s	13122	3.11s
nql-9-6	NR	>2000s	NR	>2000s	NR	>2000s	39366	9.11s
nql-10-6	NR	>2000s	NR	>2000s	NR	>2000s	118098	27.86s
nql-3-8	128	30.00ms	128	70.00ms	128	1.09s	128	31.00ms
nql-4-8	NR	>2000s	NR	>2000s	512	33.28s	512	78.00ms
nql-5-8	NR	> 2000s	NR	>2000s	NR	>2000s	2048	281.00ms
nql-6-8	NR	> 2000s	NR	>2000s	NR	>2000s	8192	1.11s
nql-7-8	NR	>2000s	NR	>2000s	NR	>2000s	32768	4.33s
nql-8-8	NR	>2000s	NR	>2000s	NR	>2000s	131072	20.39s
nql-9-8	NR	> 2000s	NR	>2000s	NR	>2000s	524288	92.62s
nql-10-8	NR	> 2000s	NR	>2000s	NR	>2000s	2097152	5.87m

Experiments (6/6)

Table: `iMult` and `IntersectionMultiplicity` on simple-nql-*n*-*d*

	iMult				IntersectionMultiplicity			
Order	$x_1 \succ \ldots \succ x_n$		$x_1 \prec \ldots \prec x_n$		$x_1 \succ \ldots \succ x_n$		$x_1 \prec \ldots \prec x_n$	
System	Im	CPU Time	Im	CPU Time	Im	CPU Time	Im	CPU Time
simple-nql-4-4	256	<1.00ms	256	<1.00ms	256	547.00ms	256	63.00ms
simple-nql-5-4	1024	<1.00ms	1024	<1.00ms	1024	3.09s	1024	250.00ms
simple-nql-6-4	4096	10.00ms	4096	10.00ms	4096	16.62s	4096	437.00ms
simple-nql-7-4	16384	250.00ms	16384	200.ms	16384	79.55s	16384	2.05s
simple-nql-8-4	NR	>2000s	NR	>2000s	NR	ERROR	65536	7.50s
simple-nql-4-8	4096	<1.00ms	4096	<1.00ms	4096	8.16s	4096	219.00ms
simple-nql-5-8	32768	180ms	32768	160.00ms	32768	93.30s	32768	1.56s
simple-nql-6-8	262144	13.78s	262144	13.65s	NR	ERROR	262144	12.69s
simple-nql-7-8	NR	>2000s	NR	>2000s	NR	ERROR	12097152	99.95s
simple-nql-8-8	NR	>2000s	NR	>2000s	NR	ERROR	16777216	12.18m

Polynomial Systems

- See [3, 4, 2, 9] for descriptions of the polynomial systems studied.
- Some systems were modified from their original presentation, such modifications are described further in the corresponding manuscript.

Table of Contents

Conclusion

- The generalization of Fulton's algorithm provides a viable alternative to intersection multiplicity algorithms which rely on standard bases.
- Experimentally, the generalization of Fulton's algorithm outperforms intersection multiplicity algorithms which use standard bases on large polynomial systems as well as other standard basis-free intersection multiplicity algorithms.
- By allowing points encoded by zero-dimensional regular chains as input, our modified version of the `IntersectionMultiplicity` command can compute the intersection multiplicity at any point, rational or not.
- The simplicity of the generalization of Fulton's algorithm (at a point) makes it easily implementable in almost any computer algebra system.

References I

Parisa Alvandi, Marc Moreno Maza, Éric Schost, and Paul Vrbik.
A standard basis free algorithm for computing the tangent cones of a space curve.
In Vladimir P. Gerdt, Wolfram Koepf, Werner M. Seiler, and Evgenii V. Vorozhtsov, editors, *Computer Algebra in Scientific Computing*, pages 45–60, Cham, 2015. Springer International Publishing.

Dario A. Bini and Bernard Mourrain.
Polynomial test suite.
`http://www-sop.inria.fr/saga/POL/`.
Accessed: 2022.

References II

François Boulier, Changbo Chen, François Lemaire, and Marc Moreno Maza.
Real root isolation of regular chains.
In Ruyong Feng, Wen-shin Lee, and Yosuke Sato, editors, *Computer Mathematics, 9th Asian Symposium (ASCM 2009), Fukuoka, Japan, December 2009, 10th Asian Symposium (ASCM 2012), Beijing, China, October 2012, Contributed Papers and Invited Talks*, pages 33–48. Springer, 2009.

Barry H. Dayton and Zhonggang Zeng.
Computing the multiplicity structure in solving polynomial systems.
In Manuel Kauers, editor, *Symbolic and Algebraic Computation, International Symposium ISSAC 2005, Beijing, China, July 24-27, 2005, Proceedings*, pages 116–123. ACM, 2005.

References III


William Fulton.
Algebraic curves - an introduction to algebraic geometry (reprint vrom 1969).
Advanced book classics. Addison-Wesley, 1989.


Gert-Martin Greuel, Santiago Laplagne, and Gerhard Pfister.
Singular manual, imult.
`https://www.singular.uni-kl.de/Manual/4-0-3/sing_1277.htm`.
Accessed: 2022.


Gert-Martin Greuel and Gerhard Pfister.
A Singular introduction to commutative algebra.
Springer Science & Business Media, 2012.

References IV


Steffen Marcus, Marc Moreno Maza, and Paul Vrbik.
On fulton's algorithm for computing intersection multiplicities.
In Vladimir P. Gerdt, Wolfram Koepf, Ernst W. Mayr, and Evgenii V. Vorozhtsov, editors, *Computer Algebra in Scientific Computing - 14th International Workshop, CASC 2012, Maribor, Slovenia, September 3-6, 2012. Proceedings*, volume 7442 of *Lecture Notes in Computer Science*, pages 198–211. Springer, 2012.


Jan Verschelde.
Phcpack demonstration database.
`http://homepages.math.uic.edu/~jan/demo.html`.
Accessed: 2022.


Paul Vrbik.
Computing Intersection Multiplicity via Triangular Decomposition.
PhD thesis, The University of Western Ontario, 2014.