Maple Transactions

Computing Intersection Multiplicities with Regular Chains

JÜRGEN GERHARD, Senior Director of Research, Maplesoft, Canada
MARC MORENO MAZA, Department of Computer Science, The University of Western Ontario, Canada
RYAN SANDFORD, Department of Computer Science, The University of Western Ontario, Canada

Abstract: We extend a generalization of Fulton's intersection multiplicity algorithm to handle zero-dimensional regular chains as input, allowing the generalization of Fulton's algorithm to compute intersection multiplicities at points containing non-rational coordinates. Moreover, we describe the implementation of this extension in Maple, and show that the range of input systems for which intersection multiplicities can be computed has increased substantially from existing standard basis free intersection multiplicity algorithm available in Maple. Lastly, we show our implementation of the generalization of Fulton's algorithm often outperforms the existing standard basis free intersection multiplicity algorithm, typically by one to two orders of magnitude.

Recommended Reference Format:
Jürgen Gerhard, Marc Moreno Maza, and Ryan Sandford. 2021. Computing Intersection Multiplicities with Regular Chains. 1, 1 (November 2021), 19 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

When an algebraic curve or surface has a singular point, local approximation at that point by a linear space (namely a tangent space) is not possible. Consequently, other techniques must be used instead such as computing tangent cones and intersection multiplicities. In 2014, the `RegularChains` library introduced commands for such computations, based on algorithms proposed in [7, 1]. While those algorithms enhance regular chain theory with a new application, namely the study of singularities in algebraic geometry, much work remains to be done in order to perform those computations in their full generality. For the case of intersection multiplicities, the support provided by the sub-package `AlgebraicGeometryTools` of the `RegularChains` library is essentially limited to the case of planar curves, following the ideas of William Fulton, from his textbook *Algebraic Curves* [5]. Those ideas replace the computation of standard bases (or Gröbner bases) by the manipulation of regular sequences and regular chains. The intensive experimentation reported in [9] shows that this is a very promising project. In [8], the authors provide a procedure which partially extends Fulton's intersection multiplicity algorithm to the general case. This procedure leads to a novel, standard basis free approach for computing intersection multiplicities beyond the case of two planar curves, which can cover cases the current standard basis free techniques cannot.
In this paper, we report on new developments (features and efficiency improvements) for the sub-package `AlgebraicGeometryTools` based on the work reported in [8]. First, the algorithm presented in [8], which computes the intersection multiplicity of an input polynomial system at a point, is extended in Section 4 so as to compute the intersection multiplicity of an input polynomial system at a finite set of points given by a regular chain. This extension is motivated by the fact

This work was supported by Mitacs through the Mitacs accelerate program.

Authors' addresses: Jürgen Gerhard, Senior Director of Research, Maplesoft, 615 Kumpf Dr, Waterloo, Canada, jgerhard@maplesoft.com; Marc Moreno Maza, Department of Computer Science, The University of Western Ontario, 1151 Richmond St, London, Canada, moreno@csd.uwo.ca; Ryan Sandford, Department of Computer Science, The University of Western Ontario, 1151 Richmond St, London, Canada, rsandfo@uwo.ca.

https://doi.org/10.1145/nnnnnnn.nnnnnnn

that, in practice, it is convenient to describe the solution set of a polynomial system by a set of regular chains.
Second, the range of input systems for which intersection multiplicities (at an arbitrary point or an arbitrary zero-dimensional regular chain) can be computed has increased substantially, see Section 6. Indeed, in the 2014 version of `AlgebraicGeometryTools`, input systems in dimension higher than two were handled by a reduction to the planar case requiring hypotheses that do not hold generically. With the new version of `AlgebraicGeometryTools`, based on [8], this reduction instead serves as an accessory to the adaptation of Fulton's techniques to higher dimension. While some corner cases still resist, many more input systems from the literature can now be processed.
Third, when both versions (the 2014 version and the latest one) both succeed in computing the intersection multiplicity of an input polynomial system at some point, the latest version does it faster, typically by one or two orders of magnitude, see Section 6.

2 Preliminaries

2.1 Notation and Basic Definitions

Let $\overline{\mathbb{K}}$ be an algebraically closed field. Let $\mathbb{A}^n$ denote $\mathbb{A}^n(\overline{\mathbb{K}})$, the affine space of dimension n over $\overline{\mathbb{K}}$. We define the degree of the zero polynomial to be $-\infty$ with respect to any variable. If I is an ideal of $\overline{\mathbb{K}}[x_1, \ldots, x_n]$, we denote by $\mathbf{V}(I)$ the algebraic set (aka variety) consisting of the common zeros to all polynomials in I. An algebraic set $\mathbf{V}$ is irreducible, whenever $\mathbf{V} = \mathbf{V}_1 \cup \mathbf{V}_2$ for some algebraic sets $\mathbf{V}_1, \mathbf{V}_2$, implies $\mathbf{V} = \mathbf{V}_1$ or $\mathbf{V} = \mathbf{V}_2$. Let $p \in \mathbb{A}^n$ be a point. For polynomials $f_1, \ldots, f_n \in \overline{\mathbb{K}}[x_1, \ldots, x_n]$, we say $\mathbf{V}(f_1), \ldots, \mathbf{V}(f_n)$ have a common component which passes through p if when we write $\mathbf{V}(f_1, \ldots, f_n)$ as a union of its irreducible components, say $\mathbf{V}_1 \cup \ldots \cup \mathbf{V}_m$, one component $\mathbf{V}_i$ contains p. For convenience, we also say that $f_1, \ldots, f_n$ have a common component through p when $\mathbf{V}(f_1), \ldots, \mathbf{V}(f_n)$ have a common component which passes through p. Recall that an algebraic set $V \subseteq \mathbb{A}^n$ is zero-dimensional if it contains only finitely many points of $\mathbb{A}^n$. In this section, all ideals are in the local ring at p, where p is often implicitly given.

Definition 2.1 (Local Ring). We define the local ring at p as

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

Local rings have a unique maximal ideal. For $\mathcal{O}_{\mathbb{A}^n,p}$, all elements which vanish on p are in the maximal ideal and all of those that do not are units. Hence, given an element $f \in \overline{\mathbb{K}}[x_1, \ldots, x_n]$ we can test whether f is invertible in $\mathcal{O}_{\mathbb{A}^n,p}$ by testing $f(p) \neq 0$.

Definition 2.2 (Intersection Multiplicity). Let $f_1, \ldots, f_n \in \overline{\mathbb{K}}[x_1, \ldots, x_n]$. We define the intersection multiplicity of $f_1, \ldots, f_n$ at $p \in \mathbb{A}^n$ 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 $\overline{\mathbb{K}}$. That is,

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

An important tool used in the extension of Fulton's algorithm is that of the regular sequence. The authors of [8] require the input system of polynomials form a regular sequence in the local ring at the point of interest. Regular sequences in Noetherian local rings enjoy several nice properties described in [6, Section 3-1], namely, in such a case, being a regular sequence is invariant under permutation. In light of this, we give a simplified definition of a regular sequence, since all regular sequences discussed in this paper will satisfy the above constraint.

Definition 2.3 (Regular Sequence). When $f_1, \ldots, f_n \in \overline{\mathbb{K}}[x_1, \ldots, x_n]$ generate a proper ideal in the local ring at p, we say $f_1, \ldots, f_n$ is a regular sequence if no f_i is zero or a zero divisor modulo $\langle f_1, \ldots, \widehat{f_i}, \ldots, f_n \rangle$, where $\widehat{f_i}$ denotes the omission of f_i.

Often, we will simply say $f_1, \ldots, f_n$ is a regular sequence when the point p is implicit.

2.2 Fulton's Algorithm

It is shown in [5, Section 3-3] that the following seven properties characterize intersection multiplicity of bivariate curves. Moreover, these seven properties lead to an algorithmic construction which computes the intersection multiplicity of a pair of bivariate curves, see Algorithm 1.

Proposition 2.4 (Fulton's Properties). *Let $p = (p_1, p_2) \in \mathbb{A}^2$ and $f, g \in \overline{\mathbb{K}}[x, y]$.*

(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, otherwise* $\mathrm{IM}(p;\ f, g) = \infty$.
(2-2) $\mathrm{IM}(p;\ f, g) = 0$ *if and only if* $p \notin \mathbf{V}(f) \cap \mathbf{V}(g)$.
(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) $\mathrm{IM}(p;\ f, g) \geq m_f m_g$ *where m_f and m_g are the respective tailing degrees of f and g expressed in* $\overline{\mathbb{K}}[x - p_1, y - p_2]$. *Moreover,* $\mathrm{IM}(p;\ f, g) = m_f m_g$ *when* $\mathbf{V}(f)$ *and* $\mathbf{V}(g)$ *intersect transversally, i.e. have no tangent lines in common.*
(2-6) $\mathrm{IM}(p;\ f, gh) = \mathrm{IM}(p;\ f, g) + \mathrm{IM}(p;\ f, h)$ *for any* $h \in \overline{\mathbb{K}}[x, y]$.
(2-7) $\mathrm{IM}(p;\ f, g) = \mathrm{IM}(p;\ f, g + hf)$ *for any* $h \in \overline{\mathbb{K}}[x, y]$.

Remark 1. *We use uppercase* IM *to denote intersection multiplicities and lowercase* im *to denote the procedures used to compute intersection multiplicities.*

Algorithm 1: Fulton's algorithm

```
1  Function im(p; f, g)
       Input: Let: x ≻ y
       (1) p ∈ 𝔸² the origin.
       (2) f, g ∈ 𝕂̄[x, y] such that gcd(f, g)(p) ≠ 0.
       Output: IM(p; f, g)
2      if f(p) ≠ 0 or g(p) ≠ 0 then                          /* Red */
3          return 0
4      r := deg_x (f(x, 0))
5      s := deg_x (g(x, 0))
6      if r > s then                                         /* Green */
7          return im(p; g, f)
8      if r < 0 then                                         /* y | f, Yellow */
9          write g(x, 0) = x^m(a_m + a_{m+1}x + ...)
10         return m + im(p; quo(f, y; y), g)
11     else                                                  /* Blue */
12         g' := lc(f(x, 0))g − (x)^{s−r} lc(g(x, 0))f
13         return im(p; f, g')
```

The following proposition was proved by Fulton in [5, Section 3-3]. It is included here for the reader's convenience.

Proposition 2.5. *Algorithm 1 is correct and terminates.*

Proof. By (2-3) we may assume p is the origin. Let f, g be polynomials in $\overline{\mathbb{K}}[x, y]$ with no common component through the origin. By (2-1), $\mathrm{IM}(p;\, f, g)$ is finite. We induct on $\mathrm{IM}(p;\, f, g)$ to prove termination. Suppose $\mathrm{IM}(p;\, f, g) = 0$, then by (2-2), at least one of f or g does not vanish at the origin and algorithm 1 correctly returns zero.
Now suppose $n := \mathrm{IM}(p;\, f, g) > 0$ for some $n \in \mathbb{N}$. Let r, s be the respective degrees of f, g evaluated at $(x, 0)$. By (2-4) we may reorder f, g to ensure $r \leq s$. Notice $r, s \neq 0$ since f, g vanish at the origin. If $r < 0$, then f is a univariate polynomial in y which vanishes at the origin, hence f is divisible by y. By (2-6) we have,

$$\mathrm{IM}(p;\, f, g) = \mathrm{IM}(p;\, y, g) + \mathrm{IM}(p;\, \mathrm{quo}(f, y; y), g)\,.$$

By definition of intersection multiplicity $\mathrm{IM}(p;\, y, g) = \mathrm{IM}(p;\, y, g(x, 0))$. Since $g(x, 0)$ vanishes at the origin and since g has no common component with f at the origin, $g(x, 0)$ is a non-zero univariate polynomial divisible by x. Write $g(x, 0) = x^m(a_m + a_{m+1}x + \ldots)$ for some $a_m, a_{m+1}, \ldots \in \overline{\mathbb{K}}$ where m is the largest positive integer such that $a_m \neq 0$. Applying (2-6), (2-5), and (2-2) yields

$$\mathrm{IM}(p;\, f, g) = m + \mathrm{IM}(p;\, \mathrm{quo}(f, y; y), g)\,.$$

Thus, algorithm 1 returns correctly when $r < 0$. Moreover, we can compute $\mathrm{IM}(p;\, \mathrm{quo}(f, y; y), g) < n$ by induction.
Now suppose $0 < r < s$. By (2-7), replacing g with g' preserves the intersection multiplicity. Notice such a substitution strictly decreases the degree in x of $g(x, 0)$. After finitely many iterations, we will obtain curves F, G such that $\mathrm{IM}(p;\, f, g) = \mathrm{IM}(p;\, F, G)$ and the degree in x of $F(x, 0) < 0$. ♮

2.3 The Generalization of Fulton's Algorithm

The following generalization of Fulton's properties was stated in [7] and proved in [9].

Theorem 2.6. *Let $f_1, \ldots, f_n$ be polynomials in $\overline{\mathbb{K}}[x_1, \ldots, x_n]$ such that $\mathbf{V}(f_1, \ldots f_n)$ is zero-dimensional. Let $p = (p_1, \ldots, p_n) \in \mathbb{A}^n$. Then, the intersection multiplicity $\mathrm{IM}(p;\, f_1, \ldots, f_n)$ satisfies (n-1) to (n-7) where:*

(n-1) $\mathrm{IM}(p;\, f_1, \ldots, f_n)$ *is a non-negative integer.*
(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) $\mathrm{IM}(p;\, f_1, \ldots, f_n) = \mathrm{IM}(p;\, \sigma(f_1, \ldots, f_n))$ *where* σ *is any permutation.*
(n-5) $\mathrm{IM}(p;\, (x_1 - p_1)^{m_1}, \ldots, (x_n - p_n)^{m_n}) = m_1 \cdots m_n$ *for any* $m_1, \ldots, m_n \in \mathbb{N}$.
(n-6) $\mathrm{IM}(p;\, f_1, \ldots, f_{n-1}, gh) = \mathrm{IM}(p;\, f_1, \ldots, f_{n-1}, g) + \mathrm{IM}(p;\, f_1, \ldots, f_{n-1}, h)$ *for any* $g, h \in \overline{\mathbb{K}}[x_1, \ldots, x_n]$ *such that* $f_1, \ldots, f_{n-1}, gh$ *is a regular sequence in* $O_{\mathbb{A}^n, p}$.
(n-7) $\mathrm{IM}(p;\, f_1, \ldots, f_n) = \mathrm{IM}(p;\, f_1, \ldots, f_{n-1}, f_n + g)$ *for any* $g \in \langle f_1, \ldots, f_{n-1}\rangle$.

Using these properties, the authors of [8] were able to extend Fulton's bivariate intersection multiplicity algorithm, to a partial algorithm in the n-variate setting. The algorithm does not generalize to a complete algorithm as one of the steps in Fulton's algorithm, namely the application of (2-7) on line 12, does not hold generically in the n-variate case.
Recall the values of r and s computed in lines 4-5 of algorithm 1 were used to partition Fulton's algorithm into several cases. Extending the computation of these values to the general setting is one of the key steps in the generalization of Fulton's algorithm. When generalizing this step, it quickly becomes clear that partitioning the algorithm into cases does not accurately reflect the

stucture of the procedure. Hence, to understand the behaviour of the generalized procedure, we introduce the notion of modular degrees.

Definition 2.7 (Modular Degree). Let f be a polynomial in $\overline{\mathbb{K}}[x_1, \ldots, x_n]$, $p = (p_1, \ldots, p_n) \in \mathbb{A}^n$, and x_i some variable in $\{x_1, \ldots, x_n\}$. Then the modular degree of f at p with respect to x_i is the degree in x_i of $f \mod \langle x_{i+1} - p_{i+1}, \ldots, x_n - p_n \rangle$.

When p is the origin, we denote the modular degree of f at p with respect to some x_i as $\text{moddeg}(f, x_i)$. Lemma 1 of [8] gives sufficient conditions, in terms of modular degrees, to split the intersection multiplicity computation into the sum of n strictly smaller intersection multiplicity computations. Simply put, when the matrix of modular degrees, i.e. the matrix whose i, j-th entry is the modular degree of f_i at p with respect to x_j, has all entries above the anti-diagonal equal to negative infinity, the intersection multiplicity computation can split. In light of this, the blue case and green case of Fulton's algorithm can be seen as matrix operations which strictly decrease the entries in one column of the matrix of modular degrees, and moreover reorder them to lie above the anti-diagonal.

Algorithm 2: Generalized Fulton's Algorithm

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

Input: Let: $x_1 \succ \ldots \succ x_n$.
(1) $p \in \mathbb{A}^n$ the origin.
(2) $f_1, \ldots, f_n \in \mathbb{K}[x_1, \ldots, x_n]$ such that $f_1, \ldots, f_n$ form a regular sequence in $\mathcal{O}_{\mathbb{A}^n,p}$ or 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,...,n* **then** /* Red */
  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**
    $r_j^{(i)} := \mathrm{moddeg}(f_i, x_j)$
for $j = 1, \ldots, n-1$ **do** /* Orange */
  Reorder $f_1, \ldots, f_{n-j+1}$ so that $r_j^{(1)} \leq \ldots \leq r_j^{(n-j+1)}$ /* Green */
  $m := \min(i \mid r_j^{(i)} > 0)$ or $m \leftarrow \infty$ if no such i exists
  if $m \leq (n-j)$ **then**
    for $i = m+1, \ldots, n-j+1$ **do** /* Blue */
      $d := r_j^{(i)} - r_j^{(m)}$
      $L_m := \mathrm{lc}(f_m(x_1, \ldots, x_j, 0, \ldots, 0); x_j)$
      $L_i := \mathrm{lc}(f_i(x_1, \ldots, x_j, 0, \ldots, 0); x_j)$
      if $L_m(p) \neq 0$ **then**
        $f_i' := L_m f_i - x_j^d L_i f_m$
      else if $L_m \mid L_i$ **then**
        $f_i' := f_i - x_j^d \frac{L_i}{L_m} f_m$
      else
        return Fail
    return $\mathrm{im_n}(p; f_1, \ldots, f_m, f_{m+1}', \ldots, f_{n-j+1}', \ldots, f_n)$
/* Yellow */
for $i = 1, \ldots, n-1$ **do**
  $q_i := \mathrm{quo}(f_i(x_1, \ldots, x_{n-i+1}, 0, \ldots, 0), x_{n-i+1}; x_{n-i+1})$
return
$\mathrm{im_n}(p; q_1, f_2, \ldots, f_n)$
$+ \mathrm{im}_{n-1}(p; q_2(x_1, \ldots, x_{n-1}, 0), \ldots, f_n(x_1, \ldots, x_{n-1}, 0))$
$+$
$\vdots$
$+\mathrm{im}_2(p; q_{n-1}(x_1, x_2, 0, \ldots, 0), f_n(x_1, x_2, 0, \ldots, 0))$
$+\mathrm{im}_1(p; f_n(x_1, 0, \ldots, 0))$

A full proof of algorithm 2 can be found in [8]. We will refer to the step described in lines 27 to 33 as splitting and we will refer to the recursive calls made in these lines as branches of computation, or branches for short.
The following example illustrates the rewriting process, and in particular, how it is used to obtain a matrix of modular degrees with a triangular shape.

Example 2.8. Let $f_1, f_2, f_3 \in \overline{\mathbb{K}}[x, y, z]$ be given by $f_1 = x^2, f_2 = (x+1)y + x^3, f_3 = y^2 + z + x^3$. Suppose we wish to compute the intersection multiplicity at p, the origin. The matrix of modular degrees computed in lines 6-8 of algorithm 2 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. Note that we omit the last column which corresponds to the modular degrees with respect to z, as this column has no effect on the conditions necessary to apply Lemma 1 of [8], as to split computations.
Since f_1 has minimal modular degree with respect to x, we choose f_1 as a pivot and use it to reduce the modular degrees of f_2 and f_3 with respect to x. Write

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

and

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

By (n-7) we may redefine $f_2 := f_2'$ and $f_3 := f_3'$. Hence, we consider the system given by $f_1 = x^2, f_2 = (x+1)y, f_3 = y^2 + z$. The matrix of modular degrees 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 with respect to x, we have $f_1 = (x+1)y, f_2 = y^2 + z, f_3 = x^2$, with matrix of modular degrees:

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

Since f_1 has minimal modular degree with respect to y, and moreover the leading coefficient of f_1 mod $\langle z \rangle$ is $x + 1$ which is invertible in the local ring at the origin, we may choose f_1 as a pivot and use it to reduce the modular degree of f_2 with respect to y. 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:

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

By applying Lemma 1 of [8] and splitting computations we conclude:

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

3 Regular Chains

This section is a short review of concepts from the theory of regular chains and triangular decompositions of polynomial systems. Details can be found in [2].

Assume that $\mathbb{K}$ is a perfect field. Let $\overline{\mathbb{K}}$ be the algebraic closure of $\mathbb{K}$ and $\mathbb{K}[\underline{X}]$ be the polynomial ring with over $\mathbb{K}$ and with n ordered variables $\underline{X} = X_1 \succ \ldots \succ X_n$. For $F \subseteq \mathbb{K}[\underline{X}]$, we denote by $\langle F \rangle$ and $V(F)$ the ideal generated by F in $\mathbb{K}[\underline{X}]$ and the algebraic set of $\overline{\mathbb{K}}^n$ consisting of the common roots of the polynomials of F, respectively.

3.1 Notations for polynomials

Let $a, b \in \mathbb{K}[\underline{X}]$ be polynomials, with $b \notin \mathbb{K}$. Denote by $\mathrm{mvar}(b)$, $\mathrm{init}(b)$ and $\mathrm{mdeg}(b)$ respectively, the greatest variable appearing in b (called the *main variable* of b), the leading coefficient of b w.r.t. $\mathrm{mvar}(b)$ (called the *initial* of b) and the degree of b w.r.t. $\mathrm{mvar}(b)$ (called the *main degree* of b). We denote by $\mathrm{prem}(a, b)$ and $\mathrm{pquo}(a, b)$ the pseudo-remainder and pseudo-quotient in the pseudo-division of a by b; those are, respectively the (uniquely defined) polynomials r and q such that $h^e a = qb + r$ and $r = 0$ or $\deg(r, v) < \mathrm{mdeg}(b)$ where $v = \mathrm{mvar}(b)$, $h = \mathrm{init}(b)$ and $e = \max(0, \deg(a, v) - \mathrm{mdeg}(b) + 1)$.

3.2 Triangular set

Let $T \subseteq \mathbb{K}[\underline{X}]$ be a *triangular set*, that is, a set of non-constant polynomials with pairwise distinct main variables. Denote by $\mathrm{mvar}(T)$ the set of main variables of the polynomials in T. A variable $v \in \underline{X}$ is called *algebraic* w.r.t. T if $v \in \mathrm{mvar}(T)$, otherwise it is said *free* w.r.t. T. For $v \in \mathrm{mvar}(T)$, we denote by T_v and T_v^- (resp. T_v^+) the polynomial $f \in T$ with $\mathrm{mvar}(f) = v$ and the polynomials $f \in T$ with $\mathrm{mvar}(f) < v$ (resp. $\mathrm{mvar}(f) > v$). Let h_T be the product of the initials of the polynomials of T. We denote by $\mathrm{sat}(T)$ the *saturated ideal* of T: if $T = \emptyset$ holds, then $\mathrm{sat}(T)$ is defined as the trivial ideal $\langle 0 \rangle$, otherwise it is the ideal $\langle T \rangle : h_T^\infty$. The *quasi-component* $W(T)$ of T is defined as $V(T) \setminus V(h_T)$. The Zariski closure of $W(T)$ in $\overline{\mathbb{K}}^n$, denoted by $\overline{W(T)}$, is the intersection of all algebraic sets $V \subseteq \overline{\mathbb{K}}^n$ such that $W(T) \subseteq V$ holds; moreover we have $\overline{W(T)} = V(\mathrm{sat}(T))$.

3.3 Regular chain

A triangular set $T \subseteq \mathbb{K}[\underline{X}]$ is a *regular chain* if either T is empty, or letting v be the largest variable occurring in T, the set T_v^- is a regular chain, and the initial of T_v is regular (that is, neither zero nor zero divisor) modulo $\mathrm{sat}(T_v^-)$. The *dimension* of T, denoted by $\dim(T)$, is by definition, the dimension of its saturated ideal and, as a property, equals $n - |T|$, where $|T|$ is the number of elements of T. If T has dimension zero, then T generates $\mathrm{sat}(T)$ and we have $V(T) = W(T)$. A regular chain T, is *square-free* if for all $t \in T$, the polynomial $\mathrm{der}(t)$ is regular w.r.t. $\mathrm{sat}(T)$, where $\mathrm{der}(t) = \frac{\partial t}{\partial v}$ and $v = \mathrm{mvar}(t)$. When the regular chain T is *square-free*, then the ideal $\mathrm{sat}(T)$ is radical.

3.4 Normalized regular chain

The regular chain $T \subseteq \mathbb{K}[\underline{X}]$ is said *normalized* if for every $v \in \mathrm{mvar}(T)$, none of the variables occurring in $\mathrm{init}(T_v)$ is algebraic w.r.t. T_v^-. Denote by d the dimension of T. Let $\underline{Y}$ and $\underline{U} = U_1, \ldots, U_d$

stand respectively for mvar(T) and $\underline{X} \setminus \underline{Y}$. Then, the fact that T is normalized means that for every $t \in T$ we have init(t) $\in \mathbb{K}[\underline{U}]$. It follows that if T is normalized, then T is a lexicographical Gröbner basis of the ideal that T generates in $(\mathbb{K}[\underline{U}])[\underline{Y}]$ (that is, over the field $(\mathbb{K}[\underline{U}])$ of rational functions), and we denote $\mathrm{NF}(p, T)$ the normal form a polynomial $p \in (\mathbb{K}[\underline{U}])[\underline{Y}]$ w.r.t. T as this Gröbner basis. In particular, if T is normalized and has dimension zero, then for every $t \in T$ we have init(t) $\in \mathbb{K}$.

3.5 Regular GCD

Let i be an integer with $1 \leq i \leq n$, let $T \subseteq \mathbb{K}[\underline{X}]$ be a regular chain, let $p, t \in \mathbb{K}[\underline{X}] \setminus \mathbb{K}$ be polynomials with the same main variable X_i, and $g \in \mathbb{K}$ or $g \in \mathbb{K}[\underline{X}]$ with mvar(g) $\leq X_i$. Assume that

(1) $X_i > X_j$ holds for all $X_j \in$ mvar(T), and
(2) both init(p) and init(t) are regular w.r.t. sat(T).

Denote by $\mathbb{A}$ the total ring of fractions of the residue class ring $\mathbb{K}[X_{i+1}, \ldots, X_n]/\sqrt{\mathrm{sat}(T)}$. Note that $\mathbb{A}$ is isomorphic to a direct product of fields. We say that g is a *regular GCD* of p, t w.r.t. T whenever the following conditions hold:

(G_1) the leading coefficient of g in X_i is a regular element of $\mathbb{A}$;
(G_2) g belongs to the ideal generated by p and t in $\mathbb{A}[X_i]$; and
(G_3) if $\deg(g, X_i) > 0$, then g divides both p and t in $\mathbb{A}[X_i]$, that is, both prem(p, g) and prem(t, g) belong to $\sqrt{\mathrm{sat}(T)}$.

Assume from now on that T has as many polynomials as variables. Assume also that X_i is the only variable occurring in p or t which is not algebraic in T. Therefore, the three triangular sets T, $T \cup \{p\}$ and $T \cup \{t\}$ can be regarded as zero-dimensional regular chains. Then, with this configuration, Conditions (G_1), (G_2), (G_3) imply the following properties:

(1) if $\deg(g, X_i) = 0$ holds then p is regular (actually invertible) modulo $\langle T \cup t \rangle$,
(2) if $\deg(g, X_i) > 0$ and mdeg(g) = mdeg(t) both hold, then $\sqrt{\langle T \cup t \rangle} = \sqrt{\langle T \cup g \rangle}$ holds and thus we have $V(T \cup t) = V(T \cup g)$,
(3) if $\deg(g, X_i) > 0$ and mdeg(g) < mdeg(t) both hold, let q = pquo(t, g), then $T \cup q$ is a regular chain and the following two relations hold:
   (a) $\sqrt{\langle T \cup t \rangle} = \sqrt{\langle T \cup g \rangle} \cap \sqrt{\langle T \cup q \rangle}$,
   (b) $V(T \cup t) = V(T \cup g) \cup V(T \cup q)$.

3.6 The algorithm RegularGCD

Let T, p, t be as in the previous section. In particular, we assume that T, $T \cup \{p\}$ and $T \cup \{t\}$ are zero-dimensional regular chains. Then, the function call RegularGcd(p, t, T) returns a a set of pairs $(g_1, T_1), \ldots, (g_e, T_e)$ where

(1) $T_1, \ldots, T_e \subseteq \mathbb{K}[\underline{X}]$ are regular chains such that $V(T) = V(T_1) \cup \cdots \cup V(T_e)$,
(2) $g_1, \ldots, g_e \in \mathbb{K}[\underline{X}]$ are polynomials such that for every $i = 1 \cdots e$, the polynomial g_i is a regular GCD of p, t w.r.t. T_i, and
(3) if T is square-free (resp. normalized) then all regular chains $T_1, \ldots, T_e$ are square-free (resp. normalized).

3.7 The algorithm Regularize

Let T, p be as in the previous section. The function call Regularize(p, T) computes a set of regular chains $T_1, \ldots, T_e \subseteq \mathbb{K}[\underline{X}]$ such that:

(1) for each $i = 1, \ldots, e$, either $p \in \langle T_i \rangle$ holds or p is regular w.r.t. $\langle T_i \rangle$,
(2) we have $V(T) = V(T_1) \cup \cdots \cup V(T_e)$,

(3) moreover, if T is square-free (resp. normalized) then all regular chains $T_1, \ldots, T_e$ are square-free (resp. normalized).

For $F \subseteq \mathbb{K}[\underline{X}]$, the function call RegularizeList(F, T) computes a pair of sets of regular chains $T_1, \ldots, T_e \subseteq \mathbb{K}[\underline{X}]$ such that:

(1) for each $i = 1, \ldots, e$, for each $p \in F$ either $p \in \langle T_i \rangle$ holds or p is regular w.r.t. $\langle T_i \rangle$,
(2) we have $V(T) = V(T_1) \cup \cdots \cup V(T_e)$,
(3) moreover, if T is square-free (resp. normalized) then all regular chains $T_1, \ldots, T_e$ are square-free (resp. normalized).

In practice RegularizeList(F, T) will separate the regular chains for which p is regular from those which generate an ideal containing p, for each $p \in F$. That is, if RegularizeList(F, T) returns a pair U, V, we will let U denote the set of regular chains for which all $p \in F$ are regular and V the set of regular chains for which $p \in \langle T_i \rangle$ for all $T_i \in V$ and $p \in \langle F \rangle$.

3.8 Triangular decomposition

Let $F \subseteq \mathbb{K}[\underline{X}]$. Regular chains $T_1, \ldots, T_e$ of $\mathbb{K}[\underline{X}]$ form a *triangular decomposition* of $V(F)$ in the sense of Kalkbrener (resp. Wu and Lazard) whenever we have $V(F) = \cup_{i=1}^{e} \overline{W(T_i)}$ (resp. $V(F) = \cup_{i=1}^{e} W(T_i)$). Hence, a triangular decomposition of $V(F)$ in the sense of Wu and Lazard is necessarily a triangular decomposition of $V(F)$ in the sense of Kalkbrener, while the converse is not true.

3.9 Computing Intersection Multiplicities with Regular Chains

A standard basis free algorithm to compute intersection multiplicities using regular chains was investigated in [1, 7] by Steffen Marcus, Marc Moreno Maza, and Paul Vrbik, with a full description given in the PhD thesis of Vrbik [9]. By applying a reduction criterion based on tangent cones, the algorithm seeks to reduce a system of n polynomials in n variables to a system with $n - 1$ polynomials in $n - 1$ variables, while preserving the intersection multiplicity. Upon repeated, successful applications of the criterion, one can reduce a system to the bivariate case where Fulton's algorithm may be applied. This algorithm was implemented in *Maple* in the `IntersectionMultiplicity` command.

4 Fulton's Algorithm and It's Generalization Using Regular Chains

Both Fulton's algorithm and its generalization assume the point p is the origin. When p is rational, both algorithms can easily be adapted to handle this case directly rather than applying property (n-3) and performing an affine change of coordinates. When p is not rational, encoding p symbolically presents practical challenges that should be addressed in our implementation of the generalization of Fulton's algorithm.

One natural way of encoding these non-rational points, is as a solution to a system of polynomial equations. This can be done using a zero-dimensional regular chain to encode the point of interest. Since intersection multiplicity is defined for a point p, we must explain what it means to compute the intersection multiplicity at a zero-dimensional regular chain. Namely, since a zero-dimensional regular chain can be thought of as encoding a finite group of points in its vanishing set, we are in a sense, defining what it means to compute the intersection multiplicity at a group of points.

Definition 4.1 (Intersection Multiplicity at a Regular Chain). Let $f_1, \ldots, f_n \in \mathbb{K}[x_1, \ldots, x_n]$ and $T \subset \mathbb{K}[x_1, \ldots, x_N]$ a zero-dimensional regular chain where $N \geq n$. If $N = n$ we say $\mathrm{IM}(rc;\, f_1, \ldots, f_n) := m$, if $\mathrm{IM}(p;\, f_1, \ldots, f_n) = m$ for every $p \in \mathbf{V}(T)$, where $m \in \mathbb{N} \cup \{\infty\}$. If $N > n$ we say $\mathrm{IM}(rc;\, f_1, \ldots, f_n) := m$, if $\mathrm{IM}\left(p;\, T_{x_N}, \ldots, T_{x_{n+1}}, f_1, \ldots, f_n\right) = m$ for every $p \in \mathbf{V}(T)$, where $m \in \mathbb{N} \cup \{\infty\}$.

Similarly, in order to extend the generalization of Fulton's algorithm to handle a zero-dimensional regular chain as input, rather than a point, we must redefine the notion of modular degree with respect to a regular chain.

Definition 4.2 (Modular Degree at a Regular Chain). Let f be a polynomial in $\overline{\mathbb{K}}[x_1, \ldots, x_n]$ and $T \subseteq \overline{\mathbb{K}}[x_1, \ldots, x_n]$ a strongly normalized, squarefree, zero-dimensional regular chain with variable ordering $x_1 \succ \ldots \succ x_n$. Suppose $\mathrm{lc}(\mathrm{NF}(f, T_{x_i}^{-}); x_i)$ is regular modulo T for some x_i. Then the modular degree of f at T with respect to x_i is the degree in x_i of $\mathrm{NF}(f, T_{x_i}^{-})$.

We shall explain why Algorithm 3 together with Algorithm 4 form a generalization of Algorithm 2 from intersection multiplicity at a point to intersection multiplicity at a (zero-dimensional) regular chain. One short explanation would be invoking the celebrated D5 Principle [4]. But, since Algorithm 2 may already split computations, more details are needed to convince the reader.
We first observe that the specifications of Algorithm 3 generalize that of Algorithm 2, thanks to Definition 4.1. For the pseudo-code, we will explain below how each key sequence of lines of Algorithm 2 is adapted to Algorithm 3.

Lines 2-3: In the regular-chain adaptation, Lines 2-5 of Algorithm 3, one must separate the points of $V(T)$ at which all polynomials $f_1, \ldots, f_n$ vanish from those at which one of $f_1, \ldots, f_n$ does not vanish; this task is achieved with RegularizeList(W, T), see Section 3.7; the construction of the set W can be seen as an optimization: indeed if all $f_1, \ldots, f_n$ have a null normal form w.r.t. T then all $f_1, \ldots, f_n$ vanish at every point of $V(T)$ and the call RegularizeList(W, T) is not needed.

Lines 4-5: these two lines in Algorithm 2 determine the trailing degree of f_1, that is, the number of times that x_1 divides f_1; in the regular-chain adaptation, Lines 6-9 of Algorithm 3, the role of x_1 is taken by T_{x_1} and the "divisibility test" is replaced by a Regular GCD computation. Because each call to RegularGCD may split the computations (thus decomposing $V(T)$) we have dedicated an algorithm to that task, namely Algorithm 4. A complete proof of that latter algorithm follows from the properties of RegularGCD given in Section 3.5. We note that ensuring that all regular chains involved in the computations are squarefree is essential: indeed, at Line 13 of Algorithm 4, we need to make sure that the division of p by g removes once and only once every root common to p and T_{x_1}, which allows us at Line 14 to increment by 1 the current value of m.

Lines 6-8: In the regular-chain adaptation, Lines 10-17 of Algorithm 3, one needs to compute modular degrees in the sense of Definition 4.2. Indeed, the leading (or trailing) degree of a polynomial w.r.t. to some variable at a regular chain must be the same at every point solution of that regular chain. This explains the call RegularizeList(C, T) together with the test $|U|+|V|> 1$; indeed, if $|U|+|V|> 1$ holds then there exists a polynomial in the list C which vanishes at some points of $V(T)$ while not vanishing at the others, that is, one modular degree is not well-defined, an issue which is resolved by splitting the computations at Line 17 of Algorithm 3.

Lines 9-23: the regular-chain adaptation, Lines 18-32 of Algorithm 3, is essentially isomorphic to its counterpart in Algorithm 2; indeed, because of the work done in Lines 10-17 of Algorithm 3, no splitting (of the zero set $V(T)$) is needed.

Lines 24-33: In the regular-chain adaptation, Lines 36-50 of Algorithm 3, one must perform all required calls to $\mathrm{im}_1, \ldots, \mathrm{im}_n$ at the same regular chains in order to calculate the sums of values returned by those calls.

Algorithm 3: Generalized Fulton's Algorithm for Regular Chains

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

Input:

(1) $f_1, \ldots, f_n \in \mathbb{K}[x_1, \ldots, x_n]$ such that for each $p \in \mathbf{V}(T)$ either $f_1, \ldots, f_n$ form a regular sequence in $\mathcal{O}_{\mathbb{A}^n,p}$, or one such f_i is a unit in $\mathcal{O}_{\mathbb{A}^n,p}$.

(2) $T \subset \mathbb{K}[x_1, \ldots, x_N]$ is a zero-dimensional, squarefree, strongly normalized regular chain in variables $x_1 \succ \ldots \succ x_N$ where $N \geq n$.

Output: A set of pairs $[m_i, T_i]$ such that: (i) $\mathbf{V}(T) = \bigcup \mathbf{V}(T_i)$, and (ii) m_i is either $\mathrm{IM}(T_i;\ f_1, \ldots, f_n)$ or Fail

$W := \{f_i \mid \mathrm{NF}(f_i,\ T) \neq 0\}$

if $W \neq \emptyset$ **then**

  $U, V :=$ RegularizeList(W, T)

  return $\{[0, H] \mid H \in U\} \cup \bigcup_{H \in V} \mathrm{im}_n(H; f_1, \ldots, f_n)$ /* Red */

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

  $U :=$ Regularize$(f_1, T^-_{x_1})$

  $U' := \bigcup_{H \in U} \{T_{x_1}\} \cup H$

  return $\bigcup_{H \in U'}$ valuation(f_1, H)

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

  for $j = 1, \ldots, n-1$ **do** /* Compute modular degrees */

    $F[i][j] := \mathrm{NF}\left(f_i,\ T^-_{x_j}\right)$

    $C[(i-1)(n-1)+j] := \mathrm{lc}(F[i][j], x_j)$

    $R[i][j] := \deg_{x_j}(F[i][j])$

$U, V :=$ RegularizeList(C, T)

if $|U|+|V| > 1$ **then**

  return $\bigcup_{H \in U \cup V} \mathrm{im}_n(H; f_1, \ldots, f_n)$

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

  Reorder $f_1, \ldots, f_{n-j+1}$ so that $R[1][j] \leq \ldots \leq R[n-j+1][j]$ /* Green */

  $m := \min(i \mid R[i][j] > 0)$ or $m := \infty$ if no such i exists

  if $m \leq (n-j)$ **then**

    for $i = m+1, \ldots, n-j+1$ **do** /* Blue */

      $d := R[i][j] - R[m][j]$

      $L_m := C[(m-1)(n-1)+j]$

      $L_i := C[(i-1)(n-1)+j]$

      if $\mathrm{NF}(L_m,\ rc) \neq 0$ **then**

        $f_i' := L_m f_i - x_j^d L_i f_m$

      else if $\mathrm{NF}\left(\frac{L_i}{L_m},\ T\right) = 0$ **then**

        $f_i' := f_i - x_j^d \frac{L_i}{L_m} f_m$

      else

        return $\{[Fail, T]\}$

    return $\mathrm{im}_n(T; f_1, \ldots, f_m, f'_{m+1}, \ldots, f'_{n-j+1}, \ldots, f_n)$

```
                                                                    /* Yellow */
  tasks := im_1(T; F[n][1])
  for i = 2, ..., n − 1 do
    newTasks := ∅
    for task in tasks do        /* each task is of the form [m, H], where H is a
     regular chain */
      m, H := task
      q := quo(NF(f_{n−i+1}, H^−_{x_i}), NF(H_{x_i}, H^−_{x_i}); x_i)
      newTasks :=
       newTasks ∪ {[m + m', H'] | [m', H'] ∈ im_i(H; q, F[n − i + 2][i], ..., F[n][i])}
    tasks := newTasks
  results := ∅
  for task in tasks do
    m, H := task
    q := quo(f_1, H_{x_1}; x_n)
    results := results ∪ {[m + m', H'] | [m', H'] ∈ im_n(H; q, f_2, ..., f_n)}
  return results
```

Algorithm 4: Valuation

Function valuation(f, T)

Input:

(1) T a zero-dimensional, squarefree, strongly normalized regular chain in variables $x_1 \succ \ldots \succ x_N$ where $N \geq n$.

(2) $f \in \mathbb{K}[x_1, \ldots, x_n]$ with main variable x_1. Moreover, $\mathrm{NF}\left(f,\ T^-_{x_1}\right) \neq 0$.

Output: A set of pairs $[m_i, T_i]$ such that: (i) $\mathbf{V}(T) = \bigcup \mathbf{V}(T_i)$, and (ii)

$$m_i = \mathrm{IM}\left(T_i;\ \mathrm{NF}\left(f,\ T^-_{i,x_1}\right)\right).$$

```
  tasks := {[f, T, 0]}
  results := ∅
  while tasks ≠ ∅ do
    p, T, m := removeElem(tasks)
    L := RegularGcd(p, T_{x_1}, T^−_{x_1})
    for g, C ∈ L do
      d := deg_{x_1}(g)
      H := {T_{x_1}} ∪ C
      if d = 0 then
        results := results ∪ {[m, H]}
      else
        q := NF(pquo(p, g, x_1), C)
        tasks := tasks ∪ {[q, H, m + 1]}
  return results
```

5 Implementation

In this section provide some details on the *Maple* implementation of the generalization of Fulton's algorithm, for both a point and a regular chain as input. Moreover, we describe the integration of this implementation with the implementation of the algorithm of Vrbik's PhD thesis, in the form of a hybrid procedure, see Section 3.9.

5.1 Overloading the IntersectionMultiplicity Command

In our implementation, we overload the `IntersectionMultiplicity` command to handle several different calling sequences. The first calling sequence occurs when one wishes to compute the intersection multiplicity at a point p, but knows the system of polynomial equations forms a regular chain. In which case, the observation made in [8, Section 5] allows us to compute the intersection multiplicity immediately by means of evaluation. The second calling sequence applies the generalization of Fulton's algorithm at a point p. The third calling sequence seeks to utilize the algorithm of Vrbik et al. along side the adaptation of the generalization of Fulton's algorithm to regular chains. That is, the third calling sequence takes a regular chain and a system of polynomial equations and applies first, the generalization of Fulton's algorithm and then, upon detecting a failure, applies the algorithm of Vrbik et al.. Lastly, the final calling sequence is used to return meaningful error messages when none of the previous calling sequences are satisfied.

5.2 IntersectionMultiplicity at a Point

The second calling sequence computes the intersection multiplicity at a point p. This calling sequence takes advantage of the cache option in its helper functions to compute the image of polynomials and modular degrees efficiently. Moreover, this calling sequence provides some support for coordinates and coefficients which are not rational given by `RootOf`. Although algebraic coordinates can be handled by encoding them in a regular chain, allowing algebraic coordinates specified by `RootOf` can simplify the calling sequence in some cases. When the system of polynomial equations, or the point of interest contain non-rational coordinates, the normalizer environment variable is set to `evala` and the algorithm proceeds as expected. Reducible `RootOf` errors are caught, in which case it is recommended the user uses the calling sequence which handles regular chains.

5.3 IntersectionMultiplicity at a Regular Chain

The third calling sequence invokes a hybrid algorithm which calls first the generalization of Fulton's algorithm, and then if necessary, the algorithm of Vrbik et al.. We apply the generalization of Fulton's algorithm first in this hybrid algorithm as it is often faster and can compute more examples than the current implementation of the algorithm of Vrbik et al., as suggested by the results in the next section. It is possible however, that the algorithm of Vrbik et al. can succeed in some cases where the generalization of Fulton's algorithm fails, as we will see in the next section; hence, a hybrid algorithm which combines the two approaches is desirable. Both the generalization of Fulton's algorithm and the algorithm of Vrbik et al. can be accessed individually in the `IntersectionMultiplicity` command by setting the optional `method` keyword equal to `fulton` or `tangentcone` respectively.

5.4 Non-Regular Sequences

The input constraints for the generalization of Fulton's algorithm require that the system of polynomials, $f_1, \ldots, f_n$, is a regular sequence at the point p, or in the case of a zero-dimensional regular chain T, that $f_1, \ldots, f_n$ is a regular sequence at all points in $\mathbf{V}(T)$ (of course, this assumption

is only required when no f_i is a unit in any of the respective local rings). Testing for this constraint is not practical for a standard basis free algorithm as it requires the use of primary decomposition which relies on the computation of Gröbner bases. Moreover, this constraint is essential to the proof of termination, hence we provide several heuristics for testing for non-regular sequences in our implementation. The key observation behind the heuristics is that by applying propositions 3 and 4 from [8, Section 2], it suffices to test for a non-regular sequence in any branch of computation. As the size of the branch decreases, which occurs during the splitting stage, testing for non-regular sequences becomes easier. For example, branches of size $n = 1$ will be a non-regular sequence only when $f_n = 0$. When $n = 2$ it suffices to check $\gcd(f_1, f_2)(p) \neq 0$, as we did in Fulton's algorithm. Applying similar heuristics during the start of each recursive call allows our implementation of the generalization of Fulton's algorithm to catch many non-regular sequences and return an error indicating the input was invalid.

5.5 Changes of Coordinates

The generalization of Fulton's algorithm requires a variable ordering to be specified before runtime. Although this ordering is needed for the algorithm, it is independent of the geometry of the input system. Hence, the choice of a good or bad ordering can cause the algorithm to succeed or fail. Since it is impractical to try all possible variable orderings, we leave the choice of variable ordering up to the user. In our implementation, the variable ordering can be changed manually by the user by modifying one or more of the required parameters. Additionally, the `maxshift` option equal to $k \in \mathbb{N}$ will apply a circular left shift to the variable ordering upon detecting a failure, for up to k failures.

5.6 Pivot Selection

The implementation of the generalization of Fulton's algorithm, at both a point and regular chain, also improves upon the pivot selection process from the procedure given in [8]. In line 11 of algorithm 2 and line 20 of algorithm 3, the index m is defined as the index of the polynomial with minimal modular degree with respect to some variable. Such an m can be thought of as the index to a pivot element, as f_m may be used to reduce the modular degrees of some polynomials within the current iteration of the algorithm. It is possible however, for multiple polynomials to share the same minimal modular degree with respect to some variable. Since the success of this procedure is often dependent on the invertibility of a particular leading coefficient in the local ring, namely that given on line 15 of algorithm 2 and line 24 of algorithm 3, having multiple viable choices for a pivot element increases the algorithm's chance of succeeding.

Consider the system $xy - z, (x + 1)y, x \in \overline{\mathbb{K}}[x, y, z]$ at the origin. Both $xy - z$ and $(x + 1)y$ have modular degree 1 with respect to y. The leading coefficients of $xy - z$ and $(x + 1)y$ modulo $\langle z \rangle$ with respect to y are x and $x + 1$ respectively. The generalization of Fulton's algorithm, as presented in [8], would return `Fail` here as x is not invertible in the local ring at the origin. But clearly this need not be the case as $x + 1$ is invertible in the local ring and $(x + 1)y$ has minimal modular degree with respect to y. Hence, extending the pivot selection process as to consider all polynomials with minimal modular degree can further strengthen the generalization of Fulton's algorithm and is therefore, included in our implementation.

6 Benchmarking

In this section we benchmark the implementation of the generalization of Fulton's algorithm relative to the implementation of the algorithm of Vrbik et al. in *Maple*. All tests were run on an Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz machine with two processors.

The size column denotes the size of the square system, that is the number of variables and number of polynomials in the system. The ordering column denotes the variable ordering given to the generalization of Fulton's algorithm. Point represents the point for which we wish to compute the intersection multiplicity at and IM represents the intersection multiplicity at that point. Method 1 and Time 1 refer to the intersection multiplicity computed and CPU time elapsed from running the generalization of Fulton's algorithm. Similarly, Method 2 and Time 2 refer to the intersection multiplicity computed and CPU time elapsed from running the algorithm of Vrbik et al.. Lastly, NA denotes not available and NR denotes no response, which occurs if a command does not return a result within a reasonable amount of time.

In table 1 we compare the two algorithms on systems chosen by the authors, at a regular chain encoding the origin. Tests 1-8 consider the system $x_1, x_2^2, x_3, \ldots, x_n$ for $n = 3, 5, 7, 9, 11, 13, 15, 25$ respectively, which give us an idea of how both algorithms scale for larger input. In particular, we see the generalization of Fulton's algorithm often runs 1-2 orders of magnitude faster than the algorithm of Vrbik et al., yielding a speed up of almost 14 minutes when $n = 25$. Tests 9,10, and 11 consider the systems

$$xy - z, x^2y^3 - z, x^4 - y,$$
$$x^3, -x^6 + y^2, z^4,$$
$$zy^2, y^5 - z^2, x^5 - y^2,$$

respectively. The algorithm of Vrbik et al. either returns an error or does not return within a reasonable amount of time on these examples. Conversely, the generalization of Fulton's algorithm can compute all of these examples. Also, worth noting, the system in test 11 contains polynomials which are all singular at the origin. Since the algorithm of Vrbik et al. requires at least one polynomial to be singular in order to apply the reduction criterion, it will always fail in such cases. Hence, the generalization of Fulton's algorithm becomes particularly attractive when all polynomials are singular at the point of interest, as it relies on a different set of conditions to succeed, unrelated to the geometry of the input. In tests 1-8 we noticed a dramatic difference in performance for a relatively simple, but increasingly large, set of input systems. Test 12 illustrates that this difference becomes even more pronounced when the polynomials themselves become more complex by considering the system $x_1^2 + x_2, x_2^2 + x_3, x_3^2 + x_1^2, x_4^2 + x_5, x_5^2 + x_6, x_6^2 + x_4$. Here, we observe that the generalization of Fulton's algorithm runs in roughly the same time as test 3, the test with 7 variables, whereas the algorithm of Vrbik et al. takes approximately 50 seconds longer than it did in test 3. This suggests that both the size of the polynomial system and complexity of the polynomials themself, will cause a significant discrepancy in the performance of the two algorithms, making the generalization of Fulton's algorithm an attractive option for large or complex systems.

Table 1. `IntersectionMultiplicity` Using the Authors' Tests

System	Size	Ordering	Point	IM	Method 1	Time 1	Method 2	Time 2
test1	3	$x_1 \succ x_2 \succ \ldots$	origin	2	2	16.00ms	2	344.00ms
test2	5	$x_1 \succ x_2 \succ \ldots$	origin	2	2	31.00ms	2	2.98s
test3	7	$x_1 \succ x_2 \succ \ldots$	origin	2	2	93.00ms	2	9.22s
test4	9	$x_1 \succ x_2 \succ \ldots$	origin	2	2	187.00ms	2	22.97s
test5	11	$x_1 \succ x_2 \succ \ldots$	origin	2	2	391.00ms	2	45.44s
test6	13	$x_1 \succ x_2 \succ \ldots$	origin	2	2	640.00ms	2	80.86s
test7	15	$x_1 \succ x_2 \succ \ldots$	origin	2	2	1.17s	2	2.20m
test8	25	$x_1 \succ x_2 \succ \ldots$	origin	2	2	7.16s	2	13.83m
test9	3	$x \succ y \succ z$	origin	5	5	31.00ms	NR	NA
test10	3	$x \succ y \succ z$	origin	24	24	109.00ms	ERROR	16.00ms
test11	3	$x \succ y \succ z$	origin	45	45	63.00ms	ERROR	15.00ms
test12	6	$x_1 \succ x_2 \succ \ldots$	origin	2	2	109.00ms	2	59.38s

In table 2 we consider the systems described in [3] at regular chains encoding just a point. In the cases where both algorithms succeed, we observe a speedup of 1-4 seconds. Moreover, in both mth191 and DZ2 we see the generalization of Fulton's algorithm is able to compute the intersection multiplicity whereas the algorithm of Vrbik et al. cannot. Lastly, in Ojika4, we see an example where the algorithm of Vrbik et al. succeeds and the generalization of Fulton's algorithm does not, which justifies the implementation of a hybrid algorithm, combining the two approaches. Although, it is worth noting that upon selecting a better variable ordering, the generalization of Fulton's algorithm can compute the intersection multiplicity of all points in Ojika4.

Table 2. `IntersectionMultiplicity` Using Examples from the Literature [3]

System	Size	Ordering	Point	IM	Method 1	Time 1	Method 2	Time 2
cbms1	3	$x \succ y \succ z$	$(0, 0, 0)$	11	FAIL	16.00ms	ERROR	16.00ms
cbms2	3	$x \succ y \succ z$	$(0, 0, 0)$	8	FAIL	31.00ms	ERROR	15.00ms
mth191	3	$x \succ y \succ z$	$(0, 1, 0)$	4	4	63.00ms	ERROR	1.42s
decker2	2	$x \succ y$	$(0, 0)$	4	4	31.00ms	4	156.00ms
Ojika2	3	$x \succ y \succ z$	$(0, 0, 1)$	2	2	47.00ms	2	1.58s
Ojika2	3	$x \succ y \succ z$	$(1, 0, 0)$	2	2	47.00ms	2	1.56s
Ojika3	3	$x \succ y \succ z$	$(0, 0, 1)$	4	4	47.00ms	4	2.24s
Ojika3	3	$x \succ y \succ z$	$(-\frac{5}{2}, \frac{5}{2}, 1)$	2	2	46.00ms	2	1.38s
Ojika4	3	$x \succ y \succ z$	$(0, 0, 1)$	3	FAIL	16.00ms	ERROR	672.00ms
Ojika4	3	$x \succ y \succ z$	$(0, 0, 10)$	3	FAIL	16.00ms	3	2.39s
Ojika4	3	$x \succ z \succ y$	$(0, 0, 1)$	3	3	47.00ms	ERROR	2.44s
Ojika4	3	$x \succ z \succ y$	$(0, 0, 10)$	3	3	79.00ms	3	4.18s
Caprasse	4	$x_1 \succ x_2 \succ \ldots$	$(2, -i\sqrt{3}, 2, i\sqrt{3})$	4	FAIL	63.00ms	NR	NA
KSS	5	$x_1 \succ x_2 \succ \ldots$	$(1, 1, 1, 1, 1)$	16	FAIL	47.00ms	ERROR	50.56s
DZ1	4	$x_1 \succ x_2 \succ \ldots$	$(0, 0, 0, 0)$	131	FAIL	16.00ms	ERROR	16.00ms
DZ2	3	$x \succ z \succ y$	$(0, 0, -1)$	16	16	94.00ms	ERROR	16.00ms

As mentioned earlier our implementation of the generalization of Fulton's algorithm can compute the intersection multiplicity of a group of points encoded by a zero-dimensional regular chain. So far,

we have only benchmarked examples where the regular chain encodes a single point. This is because the implementation of the algorithm of Vrbik et al. does not provide sufficient support for regular chains encoding a group of points and may throw an error or return an incorrect intersection multiplicity when this is the case. Instead, it is suggested to use the `TriangularizeWithMultiplicity` command to compute the intersection multiplicity of a group of points.
The `TriangularizeWithMultiplicity` command first solves the system of polynomial equations using `Triangularize` and then computes the intersection multiplicity of each of the solutions using the algorithm of Vbrik et al.. In practice, this functionality is desirable when the solutions to the system of polynomial equations are not known beforehand. In our implementation, we have extended `TriangularizeWithMultiplicity` to use the generalization of Fulton's algorithm, and hence, we provide benchmarking for `TriangularizeWithMultiplicity` as well.
Since `TriangularizeWithMultiplicity` may return several regular chains and their corresponding intersection multiplicities, we define two new columns, Success Ratio 1 and Success Ratio 2, to be the number of intersection multiplicities successfully computed over the number of regular chains returned, using the generalization of Fulton's algorithm and the algorithm of Vrbik et al. respectively. We also note that the implementation of the algorithm of Vrbik et al. returns an error any time it cannot compute all intersection multiplicities, hence Success Ratio 2 will contain only full fractions, errors, and NR.

Table 3. `TriangularizeWithMultiplicity` Using Examples from the Literature [3]

System	Size	Ordering	Success Ratio 1	Time 1	Succeess Ratio 2	Time 2
cbms1	3	$x \succ y \succ z$	10/11	656.00ms	ERROR	219.00ms
cbms2	3	$x \succ y \succ z$	1/2	12.05s	ERROR	297.00ms
mth191	3	$x \succ y \succ z$	8/8	547.00ms	ERROR	1.33s
decker2	2	$x \succ y$	3/3	46.00ms	3/3	181.00ms
Ojika2	3	$x \succ y \succ z$	4/4	187.00ms	4/4	4.71s
Ojika3	3	$x \succ y \succ z$	2/2	94.00ms	2/2	2.52s
Ojika4	3	$x \succ y \succ z$	3/5	375.00ms	ERROR	735.00ms
Ojika4	3	$x \succ z \succ y$	5/5	422.00ms	ERROR	2.70s
Caprasse	4	$x_1 \succ x_2 \succ \ldots$	NR	NA	NR	NA
KSS	5	$x_1 \succ x_2 \succ \ldots$	16/17	3.22s	ERROR	54.05s
DZ1	4	$x_1 \succ x_2 \succ \ldots$	NR	NA	ERROR	313.00ms
DZ2	3	$x \succ z \succ y$	2/2	156.00ms	ERROR	31.00ms

Once more, we note that the generalization of Fulton's algorithm is able to compute the intersection multiplicity at more points than the implementation of the algorithm of Vbrik et al.; in most cases computing the intersection multiplicity at all, or almost all, solutions to the system of interest. Moreover, for Ojika2, Ojika3, and Ojika4 (with a desirable variable ordering), the generalization of Fulton's algorithm computes the intersection multiplicities at all solutions seconds faster than the algorithm of Vrbik et al..

References

[1] 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.

[2] C. Chen and M. Moreno Maza. Algorithms for computing triangular decomposition of polynomial systems. *J. Symb. Comput.*, 47(6):610–642, 2012.

[3] 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.

[4] Jean Della Dora, Claire Dicrescenzo, and Dominique Duval. About a new method for computing in algebraic number fields. In B. F. Caviness, editor, *EUROCAL '85, European Conference on Computer Algebra, Linz, Austria, April 1-3, 1985, Proceedings Volume 2: Research Contributions*, volume 204 of *Lecture Notes in Computer Science*, pages 289–290. Springer, 1985.

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

[6] Irving Kaplansky. *Commutative rings*. The University of Chicago Press, Chicago, Ill.-London, revised edition, 1974.

[7] 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.

[8] Marc Moreno Maza and Ryan Sandford. Towards extending fulton's algorithm for computing intersection multiplicities beyond the bivariate case. In François Boulier, Matthew England, Timur M. Sadykov, and Evgenii V. Vorozhtsov, editors, *Computer Algebra in Scientific Computing - 23rd International Workshop, CASC 2021, Sochi, Russia, September 13-17, 2021, Proceedings*, volume 12865 of *Lecture Notes in Computer Science*, pages 232–251. Springer, 2021.

[9] P. Vrbik. *Computing Intersection Multiplicity via Triangular Decomposition*. PhD thesis, The University of Western Ontario, 2014.