Computing Intersection Multiplicities with Regular Chains

Maple Conference 2021

Jürgen Gerhard [1], Marc Moreno Maza [2], and Ryan Sandford [2]

[1]Maplesoft

[2]Department of Computer Science, The University of Western Ontario, Canada

October 12, 2021

Table of Contents

What Are Intersection Multiplicities

- Intersection multiplicities generalize the notion of multiplicity of a root for more than one polynomial.
- When an algebraic set has a singular point, local approximation at that point by a linear space is not possible.
- When this occurs, tools like the tangent cone and intersection multiplicity allow us to understand the behaviour of the algebraic set at that singularity.
- In a sense, intersection multiplicities tell us how complex a singularity is.

Example

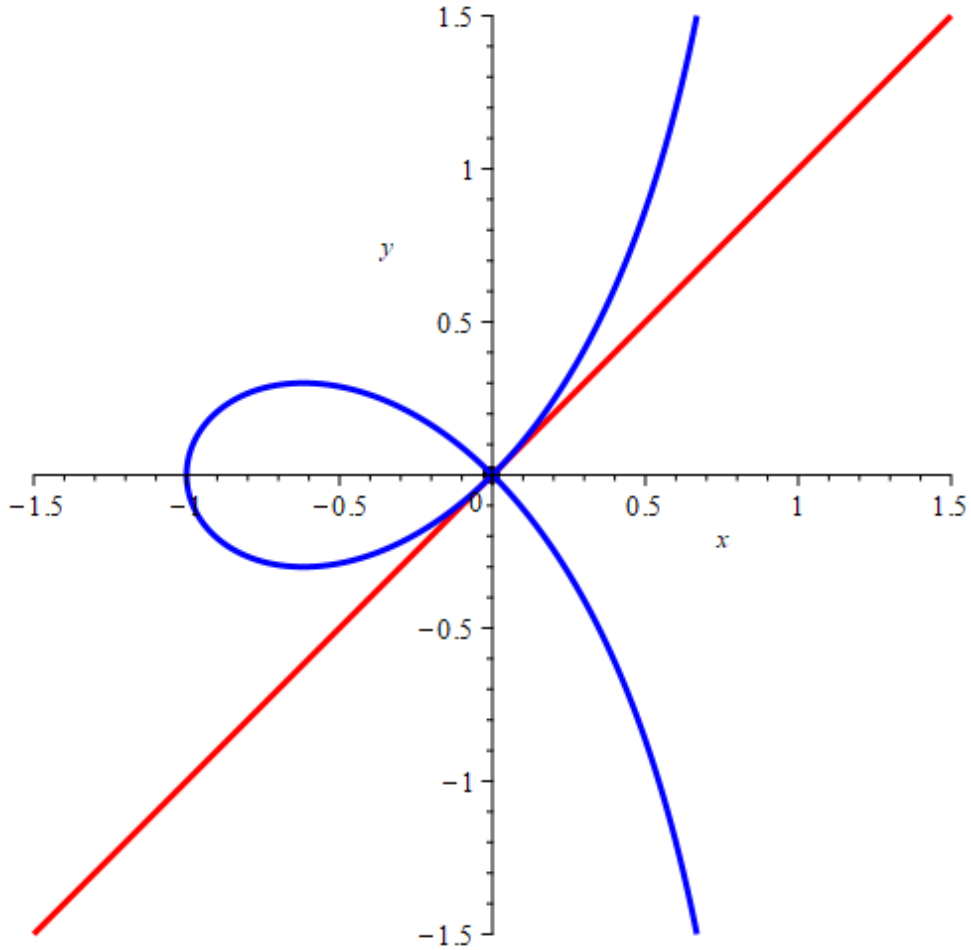

Figure: $\mathrm{IM}\left((0,0);\, y-x, x^3+xy^2+x^2-y^2\right)=3$

Intersection Multiplicities and Local Rings

- Let $\mathbb{K}$ be an algebraically closed field and let $\mathbb{A}^n$ denote affine n space over $\mathbb{K}$.

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

- In [4], Fulton presented an elegant algorithm for computing the intersection multiplicity of two planar curves by applying a set of rules to rewrite and simplify the input system.
- The first algorithmic solution in the general setting was proposed by Mora and is described in [3].
- Mora's solution is implemented in Singular and relies on the use of standard bases.
- Often, one may wish to avoid the use of standard bases. In which case, algorithms which can effectively utilize Fulton's approach in a more general setting become desirable.

Standard Basis Free Algorithms for Computing Intersection Multiplicities

- Several standard basis free approaches were investigated in CASC 2012 and 2015 [5, 1, 7], which apply an algorithmic criterion based on tangent cones to reduce to the bivariate case.
- The IntersectionMultiplicity command, introduced in Maple 2020, implemented this algorithm.
- Unfortunately, the criterion discovered in [5, 1, 7] does not always apply, and hence, the IntersectionMultiplicity command may fail.
- Recently, in CASC 2021 [6], a new standard basis free approach was investigated, where the authors present a partial algorithm extending Fulton's bivariate intersection multiplicity algorithm to the n-variate setting.

Our Contribution

- We implemented this new extended version of Fulton's algorithm in Maple in the IntersectionMultiplicity command, which now takes a hybrid approach by applying both standard basis free algorithms.
- Moreover, we further extended this algorithm to handle non-rational coordinates and benchmarked this implementation relative to the previous intersection multiplicity algorithm.

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 Chains

- For a given polynomial system F with coefficients in a field $\mathbb{K}$, the coordinates of the points in its solution set, $\mathbf{V}(F)$, may lie in a field extension of $\mathbb{K}$. Consider the system $x^2 = 2, y^2 = x + 1$ whose solution set is $\left\{(\sqrt{2}, \pm\sqrt{\sqrt{2}+1}), (-\sqrt{2}, \pm\sqrt{-\sqrt{2}+1})\right\}$.
- Therefore, computing the intersection multiplicity of F at a point, p, requires the manipulation field extensions of $\mathbb{K}$.
- In practice, these extensions do not always appear as fields but rather as direct products of fields (DPFs). Regular chains (and their implementation in the $\mathrm{RegularChains}$ library) encode such DPFs and automatically split computations when needed, in particular when a zero-divisor is encountered.

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

- In contrast, 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 $x^{1000000} = 2, y^2 = x + 1$.
- To avoid this, it is natural to encode p as a solution to a system of polynomial equations, rather then as a tuple 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.

Implementing the Generalization of Fulton's Algorithm Using Regular Chains

- In our implementation of the generalization of Fulton's algorithm, we extended the algorithm presented in [6] 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.

Table of Contents

Legend

- Size denotes the size of the square input system. That is, the number of variables/polynomials.
- Ordering denotes the variable ordering in the regular chain encoding the point. Choosing a good/bad ordering can affect the results of both algorithms.
- IM refers to the intersection multiplicity of the polynomial system at the given point.
- Method 1 and Time 1 refer to the intersection multiplicity computed by our implementation of the generalization of Fulton's algorithm and the CPU time elapsed while running this algorithm.
- Method 2 and Time 2 refer to the intersection multiplicity computed by the existing intersection multiplicity algorithm and the CPU time elapsed while running this algorithm.
- NA stands for not available and NR stands for no response. NR occurs when no result is returned within a reasonable amount of time.

Intersection Multiplicity Tests

Table: 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

- Tests 1-8 consider the polynomial system $x_1, x_2^2, x_3, \ldots, x_n$ for $n = 3, 5, 7, 9, 11, 13, 15, 25$ respectively.
- Test 9 considers $xy - z, x^2y^3 - z, x^4 - y$, test 10 uses $x^3, -x^6 + y^2, z^4$, and test 11 considers $zy^2, y^5 - z^2, x^5 - y^2$.
- Test 12 considers $x_1^2 + x_2, x_2^2 + x_3, x_3^2 + x_1^2, x_4^2 + x_5, x_5^2 + x_6, x_6^2 + x_4$.

Intersection Multiplicity Tests from the Literature

Table: IntersectionMultiplicity Using Examples from the Literature [2]

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

Triangularize With Multiplicity

- The IntersectionMultiplicity command requires an argument which specifies a solution or a subset of solutions to the polynomial system, to compute the intersection multiplicity at.
- The TriangularizeWithMultiplicity command computes the solutions to the polynomial system in the form of zero-dimensional regular chains and then calls IntersectionMultiplicity on each solution.
- Since 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 method 1 and 2 respectively.
- By nature of the implementation of method 2, an error is thrown if it is unable to compute all intersection multiplicities, hence Success Ratio 2 will always be a full fraction, an error, or NR.

Triangularize with Multiplicity Tests

Table: TriangularizeWithMultiplicity Using Examples from the Literature [2]

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

Table of Contents

Conclusion

- We implemented generalized Fulton's algorithm in Maple. Moreover, we combined this with the previous standard basis free algorithm used in the IntersectionMultiplicity command, to make a new, more powerful, hybrid version of the IntersectionMultiplicity command.
- We extended generalized Fulton's algorithm to handle a zero-dimensional regular chain as input rather than a point, allowing it to work with algebraic coordinates in practice.
- We provide benchmarking for the performance of generalized Fulton's algorithm relative to Maple's previous intersection multiplicity algorithm. The results suggest generalized Fulton's algorithm is often faster and can compute more examples.

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.

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 II

Wolfram Decker, Gert-Martin Greuel, Gerhard Pfister, and Hans Schönemann.
SINGULAR 4-1-1 — A computer algebra system for polynomial computations.
http://www.singular.uni-kl.de, 2018.

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

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*

References III

3-6, 2012. Proceedings, volume 7442 of *Lecture Notes in Computer Science*, pages 198–211. Springer, 2012.

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.

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