SIROCCO: A Library for Certified Polynomial Root Continuation

Miguel Ángel Marco-Buzunariz[1,3(✉)] and Marcos Rodríguez[2,3]

[1] Universidad of Zaragoza, Zaragoza, Spain
`mmarco@unizar.es`
[2] Centro Universitario de la Defensa de Zaragoza, Zaragoza, Spain
`marcos@unizar.es`
[3] IUMA. Instituto Universitario de Matemáticas y Aplicaciones, Zaragoza, Spain
`https://riemann.unizar.es/∼mmarco`,
`http://www.imark.es`

Abstract. The classical problem of studying the topology of a plane algebraic curve is typically handled by the computation of braid monodromies. The existence of arithmetic Zariski pairs implies that purely algebraic methods cannot provide those braids, so we need numerical methods at some point. However, numerical methods usually have the problem that floating point arithmetic introduces rounding errors that must be controlled to ensure certified results. We present SIROCCO (The source code and documentation is available in: `https://github.com/miguelmarco/sirocco`), a library for certified polynomial root continuation, specially suited for this task. It computes piecewise linear approximations of the paths followed by the roots. The library ensures that there exist disjoint tubular neighborhoods that contain both the actual path and the computed approximation. This fact proves that the braids corresponding to the approximation are equal to the ones corresponding to the actual curve. The validation is based on interval floating point arithmetic, the Interval Newton Criterion and auxiliary lemmas. We also provide a SageMath interface and auxiliary routines that perform all the needed pre and post-processing tasks. Together this is an "out of the box" solution to compute, for instance, the fundamental group of the complement of an affine complex curve.

Keywords: Validated numerics · Interval arithmetic · Homotopy · Algebraic curves

1 Introduction

The problem of studying the topology of the embedding of a curve $\mathcal{C}$ in the complex projective plane $\mathbb{CP}^2$ is a classical one in algebraic geometry. One of the most important tools in this theory is the braid monodromy. It is defined as follows. Consider $\mathcal{C}$ be a degree d curve in $\mathbb{CP}^2$. Fix a point $p \in \mathbb{CP}^2 \setminus \mathcal{C}$. The set of lines going through p forms a $\mathbb{CP}^1$. So we have a fibration

$$\begin{tikzcd}\mathbb{C} \arrow[r, hook] & \mathbb{CP}^2 \setminus \{p\} \arrow[d, "\pi"] \\ & \mathbb{CP}^1\end{tikzcd}$$

Generically, the fibres intersect the curve $\mathcal{C}$ in d distinct points. However, there is a finite set Δ of points of $\mathbb{CP}^1$ such that their preimages contains less than d points. This set Δ will be called the *discriminant* of π. Consider $Sym^d(\mathbb{C})$ the d'th symmetric product of $\mathbb{C}$. This is the configuration space of d different points in $\mathbb{C}$. It is well known that its fundamental group $\pi_1(Sym^d(\mathbb{C}))$ is the braid group in d strands B_d. As we have seen before, we have a well defined map

$$\mathbb{CP}^1 \setminus \Delta \to Sym^d(\mathbb{C})$$

which induces a map of fundamental groups

$$\pi_1(\mathbb{CP}^1 \setminus \Delta) \to B_d$$

This map is called the *braid monodromy* of the curve with respect to the projection π. It is usually presented as a list of braids, corresponding to the images of a good system of generators of $\pi_1(\mathbb{CP}^1 \setminus \Delta)$. VanKampen gave in [6] a method to compute the fundamental group of $\mathbb{CP}^2 \setminus \mathcal{C}$ from the braid monodromy. Moreover, Carmona [3] proved that the braid monodromy itself determines the topology of the pair $(\mathbb{CP}^2, \mathcal{C})$. The previous paragraphs show the interest of computing the braid monodromy. However, the existence of pairs of curves, defined by polynomials whose coefficients are Galois conjugated in some number field, but with nonhomeomorphic embeddings (the so-called Zariski pairs, see [1] for a survey on the subject), shows us that this braids cannot be computed by purely algebraic methods. In the following sections we will present a numerical (yet certified) method to compute them. Our approach will consist of computing a piecewise linear approximation of the strands of each braid. This approximation must produce the same braid as the original strands. In order to ensure this, we propose a method that certifies that the approximations live inside disjoint tubular neigbourhoods of the actual strands (See Fig. 2). The method is based on a homotopic continuation of the roots via a predictor–corrector scheme, plus a validation step using interval arithmetic and interval Newton Operator. In fact the final scheme is "predictor–validator–corrector". The predictor uses implicit differentiation and linear extrapolation. The corrector is just the classical Newton Method.

2 Validated Numerics

In the following we will need the concepts of complex interval, interval polynomial and interval evaluation.

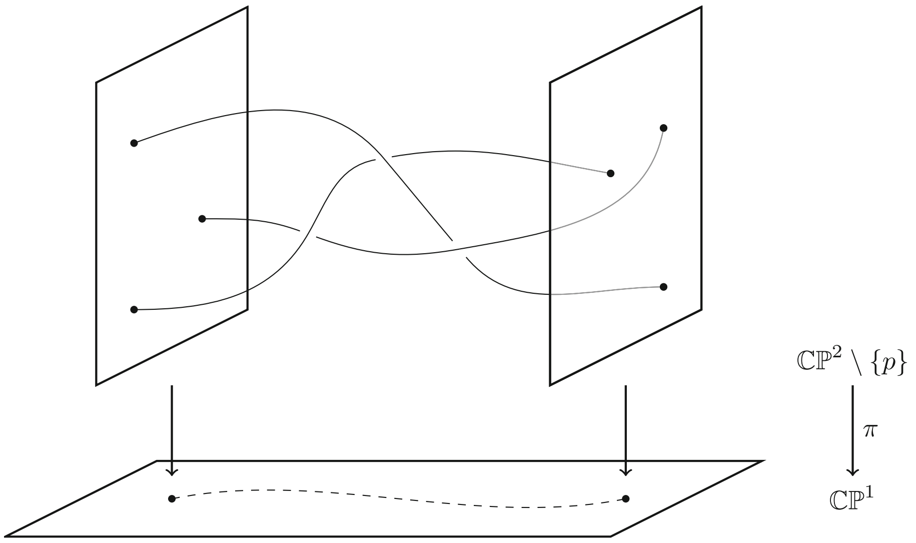

Fig. 1. Braid over a path

Definition 1. *A complex interval is a set of the form* $\{x + i \cdot y \mid a \leq x \leq b, c \leq y \leq d\}$ *for some* $a, b, c, d \in \mathbb{R}$*,* $a \leq b$*,* $c \leq d$*. The set of complex intervals will be denoted as* $I\mathbb{C}$*.*

Given $S \subseteq \mathbb{C}$, its interval closure (that is, the smallest element of $I\mathbb{C}$ that contains S) will be denoted by $[S]$. Note that complex numbers are a particular case of complex intervals. $I\mathbb{C}$ is not a ring, however, by abuse of notation, we will talk about polynomials over this set. For example, if $If = Ia_0 + Ia_1 x + \cdot Ia_n x^n \in I\mathbb{C}[x]$, it can be thought of as a way to represent the set $S(If) = \{a_0 + a_1 x + \cdots a_n x^n \in \mathbb{C}[x] \mid a_i \in Ia_i \forall i\}$.

Definition 2. *An evaluation scheme is a map*

$$E : I\mathbb{C}[x] \times I\mathbb{C} \mapsto I\mathbb{C}$$

such that for every $f \in S(If)$*, and every* $z \in Iz$*, the evaluation* $f(z)$ *is in* $E(If, Iz)$*.*

Analogously, we can define interval polynomials of two variables and their corresponding evaluation schemes. For example, the usual interval arithmetic is an evaluation scheme. From now on we will assume that we have fixed evaluation schemes E_1, E_2 for univariate and bivariate interval polynomials respectively. By abuse of notation, $E_1(If, Iz)$ will be denoted by $If(Iz)$, and $E_2(If, Ix, Iy)$ will be denoted by $If(Ix, Iy)$.

2.1 Newton Method

A basic tool to prove statements with a computer which cannot be proved in a symbolic way is the Interval Newton Method [7,9]. Among all its possible

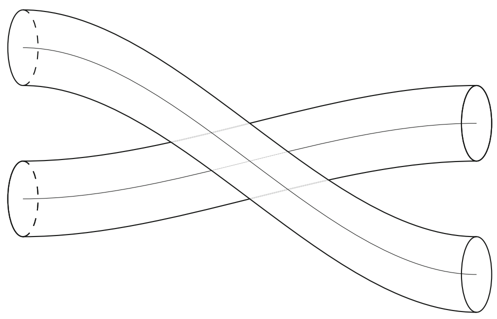

Fig. 2. Tubular neigbourhoods of the strands. The piecewise linear approximations live inside them.

formulations, we present here the one for complex univariate polynomials, since it is the one we need.

Theorem 1. *Let $f : \Omega \to \mathbb{C}$, $\Omega \subseteq \mathbb{C}$ an open set, $f \in \mathcal{C}^\infty(\Omega)$. Let $Y \in I\mathbb{C}$, $y_0 \in Y \subset \Omega$. Let us assume that $0 \notin [f'(Y)]$. We call the Interval Newton Operator:*

$$N(y_0, Y, f) = y_0 + f(y_0)/[f'(Y)].$$

Then:

- *If $y_1, y_2 \in Y$ such that $f(y_1) = f(y_2)$, then $y_1 = y_2$.*
- *If $N(y_0, Y, f) \subseteq Y$, then $\exists | y^* \in Y$ such that $f(y^*) = 0$.*
- *If $y_1 \in Y$ such that $f(y_1) = 0$, then $y \in N(y_0, Y, f)$.*
- *If $N(y_0, Y, f) \cap Y = \emptyset$, then $f(y) \neq 0$, $\forall y \in Y$.*

A detailed proof of this Theorem can be found in reference [10]. Further details on validation methods can be found in [7]. Note that, in the same way we defined an evaluation scheme for polynomials, we can also define an evaluation scheme for the Newton operator. Again, let us assume that we have fixed such an evaluation scheme.

Corollary 1 (Newton method for interval polynomials). *Consider $If \in I\mathbb{C}[x, y]$ a complex intervalar polinomial. Let $y_0 \in \mathbb{C}$, $Ix, Iy \in I\mathbb{C}$ such that $y_0 \in Iy$. Consider If_{Ix} the univariate intervalar polynomial resulting from evaluating If at $x = Ix$. If $N(y_0, Iy, If_{Ix}) \subseteq Iy$, then for every $f \in If$ and every $x \in Ix$, there exists a unique root (counted with multiplicity) of f_x in Iy. Moreover, this root lies in $N(y_0, Iy, If_{Ix})$.*

3 The SIROCCO Library

In this section we present a C library developed for the purpose described in the Introduction. The core function provided by the library is called `homotopyPath`.

It takes as input a polynomial $f(x, y)$ (as a list of its coefficients) and an approximation y_0 of a root $f(0, y)$. Its output is a list with the points that determine a good (as explained in the Introduction) approximation of the path followed by y_0 as x moves from 0 to 1. A simple change of variable allows us to translate any other linear path to this one. In the following of this section we will briefly present the method implemented in SIROCCO. First we set a trivial Lemma to ensure that the neighbourhoods we use will be disjoint. The description of the method will follow the notation set in this lemma.

Lemma 1. *Let C_1, C_2 be two concentric squares with horizontal and vertical sides, being C_2 three times bigger than C_1. Let C_1', C_2' another pair of squares with the same properties. If there exists points $x \in C_1 \setminus C_2'$ and $y \in C_1' \setminus C_2$, then C_1 and C_1' are disjoint.*

3.1 The Validated Continuation Algorithm

The basic outline of the method consists on the following: Start with a polynomial $f_0(x, y) \in \mathbb{C}[x, y]$, an interval polynomial $If(x, y) \in I\mathbb{C}[x, y]$ that contains it (can be f_0 itself), and $y_{inp} \in \mathbb{C}$ an approximate root of $f(0, y)$. Set $x_0 = 0$, $y_0 = y_{inp}$ For each step we want to compute two boxes $Ix \times IY_1$ and $Ix \times IY_2$ such that $\forall x \in Ix, \forall f \in If$, $f_x(y)$ has a unique root in IY_1 and in IY_2. IY_1 and IY_2 are both centered in y_0. IY_2 will be three times wider than IY_1 (and they will play the roles of C_1 and C_2 in Lemma 1). Then:

1. Estimate[1] an initial value $\delta > 0$ to be the radius of IY_1.
2. Apply Corollary 1 to If with $Ix = [x_0]$, $Iy = C_i$ as in Lemma 1 ($i = 1, 2$). Keep reducing δ until Corollary 1 is satisfied.
3. Estimate[2] $h > 0$ to be the stepsize for the predictor.
4. Apply Corollary 1 to If with $Ix = [x_0, x_0 + h]$, $Iy = C_i$. Keep reducing h until the Corollary is satisfied.
5. Apply classical Newton method to correct y_0 for polynomial f_{x_0+h}. Use corrected value to update y_0. Set x_0 to $x_0 + h$.

Stages 1 to 5 shall be repeated until x_0 reaches 1. In order to obtain longer steps in the validation, in stages 3 and 4 we use the following trick. We define an auxiliar polynomial.

$$g(x, y) = f(x + x_0, y + a(x + x_0))$$

This change of variables sends the point (x_0, y_0) to $(0, y_0)$ and transform the implicit curve given by $f(x, y) = 0$ into a curve whose implicit derivative at the translated point vanishes (see Fig. 3). The validation of this new polynomial in a rectangular interval box implies the validation of the original polynomial in the box transformed by the change of variables. In practice, the longer stepsizes that this trick allows compensates the computation effort of the change of variables. Experimental evidence shows an important speedup.

[1] This estimation is derived from the degree 2 Taylor expansion of the polynomial.
[2] This estimation is derived from the degree 2 Taylor expansion of the implicit function defined by $f(x, y) = 0$.

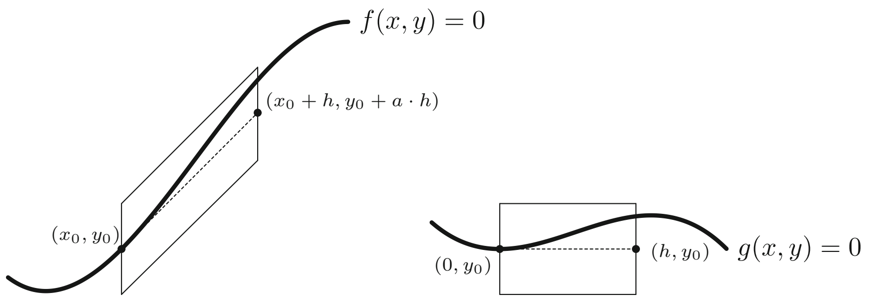

Fig. 3. Neighborhoods related through the change of variables ϕ.

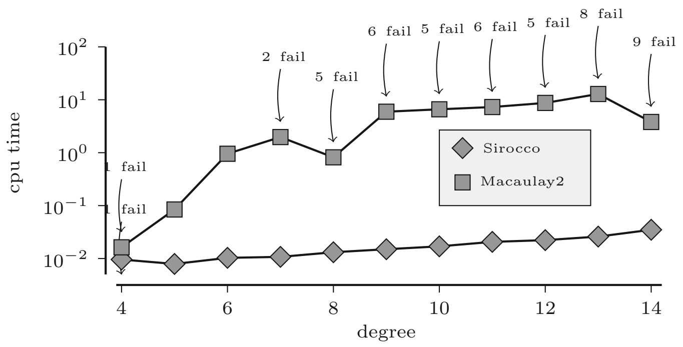

Fig. 4. Timing comparison. CPU-time vs degree of validated polynomial using softwares SIROCCO and Macaulay2

4 Comparison and Timimgs

Up to the authors's knowledge, there are several software packages able to perform validated homotopy continuation, such as `pss5` [8], `Cadenza` [5] or the `NumericalAlgebraicGeometry` package [2] of `Macaulay2` [4]. However, they have different objectives. We will now compare the performance of our implementation with the one by Leykin in the `NumericalAlgebraicGeometry` package. We remark again that the purpose of `NumericalAlgebraicGeometry` package is not to compute braid monodromies, but to find solutions of polynomial systems. In that sense this comparison is not fully fair. Our comparison consisted in timing the computation of the strand starting at one root y_0 of a polynomial $f(x,y)$ from $x=0$ to $x=1$ for several polynomials. The polynomials were chosen randomly among the polynomials of degree 4 to 14 (10 polynomials of each degree). All measurements were made by averaging 5 runs of the program on the same input. The test platform has an Intel Core i5-4570 CPU running at 3.20 GHz, with 8GB of RAM. Both softwares are configured to throw an error message when they are not able to validate a step (those timings are not taken into account in the average) All timings are expressed in seconds. In Fig. 4 we can see that our implementation is consistently faster (as we could expect) than the one in Macaulay2. Moreover, it is also more robust, since it gives the right answer in cases where Macaulay2 could not guarantee the correctness. The difference in timings varies greatly, but, on average, our implementation is about an order of magnitude faster.

References

1. Bartolo, E.A., Cogolludo, J.I., Tokunaga, H.: A survey Zariski pairs. Adv. Stud. Pure Math. **50**, 1–100 (2008)
2. Beltrán, C., Leykin, A.: Robust certified numerical homotopy tracking. Found. Comput. Math. **13**(2), 253–295 (2013)
3. Ruber, J.C.: Monodroma de trenzas de curvas algebraicas planas. Ph.D. thesis, Universidad de Zaragoza (2003)
4. Grayson, D.R., Stillman, M.E.: Macaulay2, a software system for research in algebraic geometry. http://www.math.uiuc.edu/Macaulay2/
5. Hauenstein, J.D., Haywood, I., Liddell Jr., A.C., Cadenza: certifying homotopy paths for polynomial systems. http://www.nd.edu/aliddel1/research/cadenza
6. Van Kampen, E.R.: On the fundamental group of an algebraic curve. Am. J. Math. **55**(1–4), 255–260 (1933)
7. Krawczyk, R., Neumaier, A.: An improved interval newton operator. J. Math. Anal. Appl. **118**(1), 194–207 (1986)
8. Malajovich, G.: Polynomial System Solver. https://sourceforge.net/projects/pss5/
9. Moore, R.E.: Interval Analysis, vol. 4. Prentice-Hall, Englewood Cliffs (1966)
10. Zgliczyński, P.: Interval krawczyk and newton method, February 2007, Lecture notes. http://ww2.ii.uj.edu.pl/~zgliczyn/cap07/krawczyk.pdf