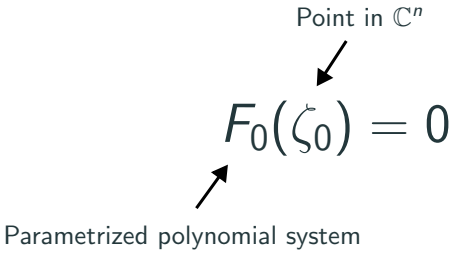
Belyi map verification using certified path tracking

Alexandre Guillemot and John Voight
MATHEXP, Inria, France

ICMS
July 21, 2026 | Wilfrid Laurier University and University of Waterloo

Certified homotopy continuation

Input. F



Certified homotopy continuation

Input. F, ζ_0

Unique continuous extension

$$F_t(\zeta_t) = 0, \quad \forall t \in [0,1]$$

Parametrized polynomial system

Certified homotopy continuation

Input. F, ζ_0

Certified homotopy continuation

Unique continuous extension

$$F_t(\zeta_t) = 0, \quad \forall t \in [0,1]$$

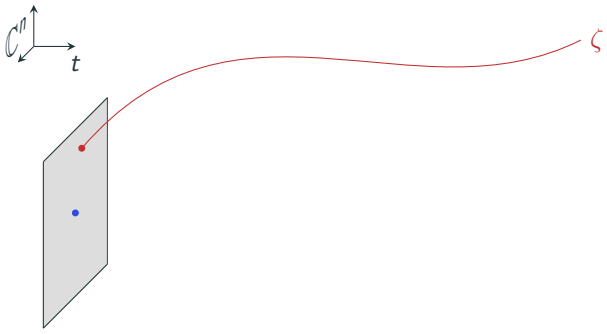
Parametrized polynomial system

Certified homotopy continuation

Input. F, ζ_0

Output. A "certified approximation" of ζ

Algorithm

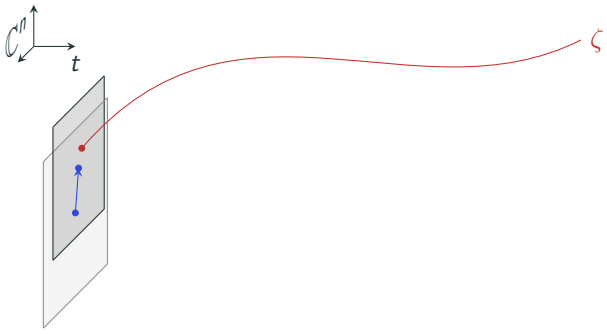


$F : \mathbb{C} \times \mathbb{C}^n \to \mathbb{C}^n$, $F_t(z) = F(t, z)$,
m isolates a zero of F_0.

def `Track`(F, m)**:**

1. $t \leftarrow 0$; $\quad L \leftarrow []$
2. **while** $t < 1$**:**
3. $\quad m \leftarrow$ `Refine`(F_t, m)
4. $\quad \delta \leftarrow$ `Extend`(F, t, m)
5. $\quad t \leftarrow t + \delta$
6. $\quad$ append (t, m) to L
7. **return** L

Algorithm

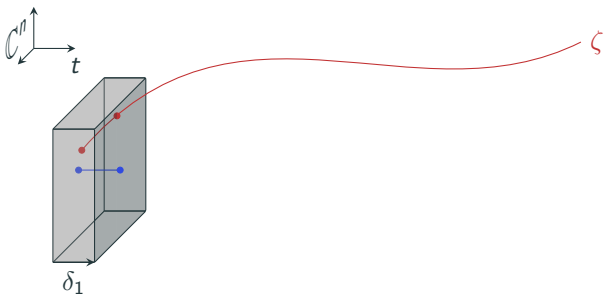


$F : \mathbb{C} \times \mathbb{C}^n \to \mathbb{C}^n$, $F_t(z) = F(t, z)$,
m isolates a zero of F_0.

def `Track`(F, m)**:**

1. $t \leftarrow 0$; $\quad L \leftarrow []$
2. **while** $t < 1$**:**
3. $\quad m \leftarrow$ `Refine`(F_t, m)
4. $\quad \delta \leftarrow$ `Extend`(F, t, m)
5. $\quad t \leftarrow t + \delta$
6. $\quad$ append (t, m) to L
7. **return** L

Algorithm

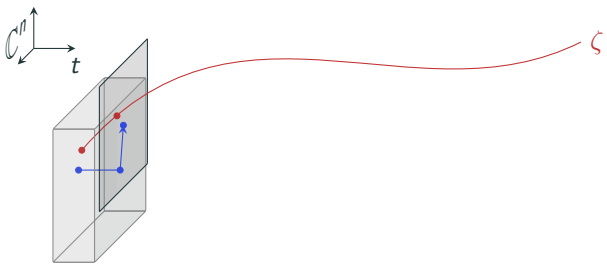


$F : \mathbb{C} \times \mathbb{C}^n \to \mathbb{C}^n$, $F_t(z) = F(t, z)$,
m isolates a zero of F_0.

def `Track`(F, m)**:**

1. $t \leftarrow 0$; $\quad L \leftarrow []$
2. **while** $t < 1$**:**
3. $\quad m \leftarrow$ `Refine`(F_t, m)
4. $\quad \delta \leftarrow$ `Extend`(F, t, m)
5. $\quad t \leftarrow t + \delta$
6. $\quad$ append (t, m) to L
7. **return** L

Algorithm

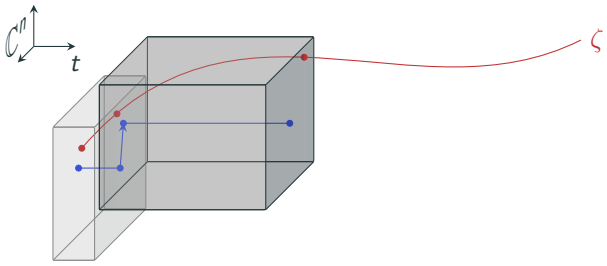


$F : \mathbb{C} \times \mathbb{C}^n \to \mathbb{C}^n$, $F_t(z) = F(t, z)$,
m isolates a zero of F_0.

def `Track`(F, m)**:**

1. $t \leftarrow 0$; $\quad L \leftarrow []$
2. **while** $t < 1$**:**
3. $\quad m \leftarrow$ `Refine`(F_t, m)
4. $\quad \delta \leftarrow$ `Extend`(F, t, m)
5. $\quad t \leftarrow t + \delta$
6. $\quad$ append (t, m) to L
7. **return** L

Algorithm

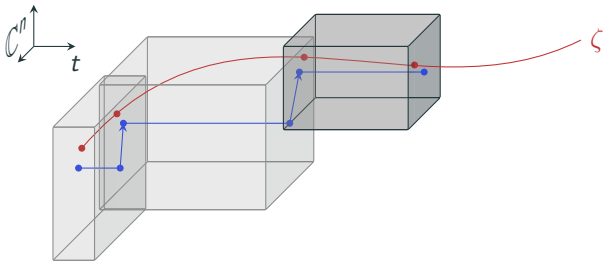


$F : \mathbb{C} \times \mathbb{C}^n \to \mathbb{C}^n$, $F_t(z) = F(t, z)$,
m isolates a zero of F_0.

def `Track`(F, m)**:**

1. $t \leftarrow 0$; $\quad L \leftarrow []$
2. **while** $t < 1$**:**
3. $\quad m \leftarrow$ `Refine`(F_t, m)
4. $\quad \delta \leftarrow$ `Extend`(F, t, m)
5. $\quad t \leftarrow t + \delta$
6. $\quad$ append (t, m) to L
7. **return** L

Algorithm



$F : \mathbb{C} \times \mathbb{C}^n \to \mathbb{C}^n$, $F_t(z) = F(t, z)$,
m isolates a zero of F_0.

def `Track`(F, m)**:**

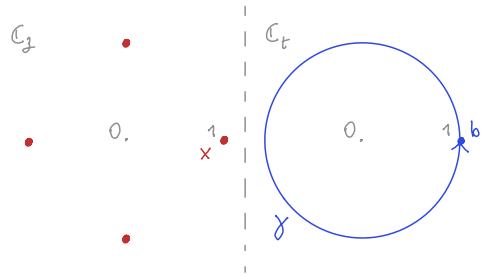
1. $t \leftarrow 0$; $\quad L \leftarrow []$
2. **while** $t < 1$**:**
3. $\quad m \leftarrow$ `Refine`(F_t, m)
4. $\quad \delta \leftarrow$ `Extend`(F, t, m)
5. $\quad t \leftarrow t + \delta$
6. $\quad$ append (t, m) to L
7. **return** L

Ingredients

- F parametrized polynomial map

  F_t: F specialized at $t \in \mathbb{C}$
- $\gamma : [0,1] \to \mathbb{C}$ loop based at $b \in \mathbb{C}$
- x zero of F_b

$$F_t(z) = z^4 - t \qquad \pi : V(F) \subset \mathbb{C}_t \times \mathbb{C}_z \to \mathbb{C}_t$$



Ingredients

- F parametrized polynomial map

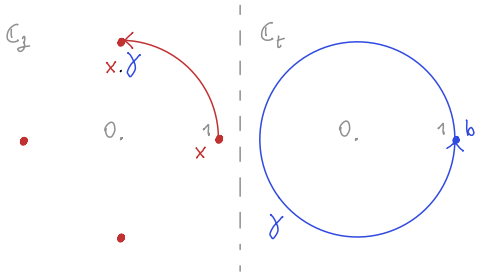
  F_t: F specialized at $t \in \mathbb{C}$
- $\gamma : [0,1] \to \mathbb{C}$ loop based at $b \in \mathbb{C}$
- x zero of F_b

Let ζ be the associated solution path.

Monodromy action/permutation

- $x \cdot \gamma := \zeta(1)$ $\quad F_{\gamma(1)}(\zeta(1)) = F_b(\zeta(1)) = 0$

$$F_t(z) = z^4 - t \qquad \pi : V(F) \subset \mathbb{C}_t \times \mathbb{C}_z \to \mathbb{C}_t$$



Ingredients

- F parametrized polynomial map

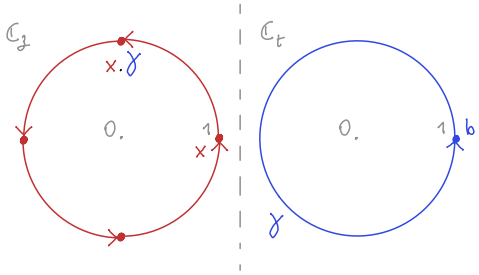
  F_t: F specialized at $t \in \mathbb{C}$
- $\gamma : [0,1] \to \mathbb{C}$ loop based at $b \in \mathbb{C}$
- x zero of F_b

Let ζ be the associated solution path.

Monodromy action/permutation

- $x \cdot \gamma := \zeta(1)$ $\quad F_{\gamma(1)}(\zeta(1)) = F_b(\zeta(1)) = 0$
- $x \mapsto x \cdot \gamma$ permutation of $F_b^{-1}(0)$

$$F_t(z) = z^4 - t \qquad \pi : V(F) \subset \mathbb{C}_t \times \mathbb{C}_z \to \mathbb{C}_t$$



Monodromy

Ingredients

- F parametrized polynomial map

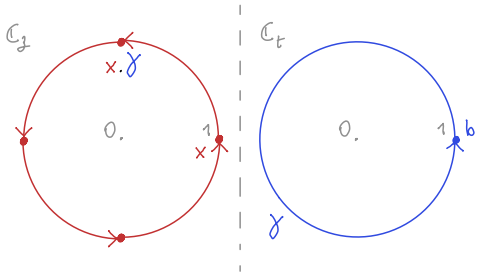
  F_t: F specialized at $t \in \mathbb{C}$
- $\gamma : [0,1] \to \mathbb{C}$ loop based at $b \in \mathbb{C}$
- x zero of F_b

Let ζ be the associated solution path.

Monodromy action/permutation

- $x \cdot \gamma := \zeta(1)$ $\quad F_{\gamma(1)}(\zeta(1)) = F_b(\zeta(1)) = 0$
- $x \mapsto x \cdot \gamma$ permutation of $F_b^{-1}(0)$

$$F_t(z) = z^4 - t \qquad \pi : V(F) \subset \mathbb{C}_t \times \mathbb{C}_z \to \mathbb{C}_t$$



Mondromy can be computed using certified homotopy continuation!

Previous work and implementations

Noncertified path trackers

- PHCpack by Verschelde (1999)
- Bertini by Bates, Sommese, Hauenstein, and Wampler (2013)
- HomotopyContinuation.jl by Breiding and Timme (2018)

Certified path trackers using Smale's alpha-theory

- NAG for M2 by Beltrán and Leykin (2012, 2013)

Certified path trackers in one variable

- Marco-Buzunariz and Rodríguez (2016)
- Kranich (2016)
- Xu, Burr, and Yap (2018)

Certified path trackers using interval arithmetic

- Kearfott and Xing (1994)
- van der Hoeven (2015) *Krawczyk operator + Taylor models*
- Duff and Lee (2024) and Guillemot and Lairez (2024)

Algpath: certified HC implementation based on Krawczyk's criterion, checked using interval arithmetic and Taylor models.

- *Rust* implementation
- available at *https://gitlab.inria.fr/numag/algpath*
- CLI, input described with a JSON file
- SLP representation of the input
- parallelized over the input fiber
- *mixed precision*, that relies on a SIMD double precision IA and Arb[1].
- automatic precision management
- **rigorous computation of monodromy permutations**

[1] Johansson, 2017.

Today's application: Belyi maps

Definition

A map $\phi : X \to \mathbb{P}^1$, where

- X is a smooth complex projective algebraic curve,
- ϕ is non-constant and unramified away from $\{0, 1, \infty\}$.

Today's application: Belyi maps

Definition

A map $\phi : X \to \mathbb{P}^1$, where

- X is a smooth complex projective algebraic curve,
- ϕ is non-constant and unramified away from $\{0, 1, \infty\}$.

Example

- Curve $X = \mathbb{P}^1$. Smooth, projective ✔

Today's application: Belyi maps

Definition

A map $\phi : X \to \mathbb{P}^1$, where

- X is a smooth complex projective algebraic curve,
- ϕ is non-constant and unramified away from $\{0, 1, \infty\}$.

Example

- Curve $X = \mathbb{P}^1$. Smooth, projective ✔
- Map ϕ: given by the univariate polynomial $f = 2x^3 - \frac{3}{2}x + \frac{1}{2}$.

Today's application: Belyi maps

Definition

A map $\phi : X \to \mathbb{P}^1$, where

- X is a smooth complex projective algebraic curve,
- ϕ is non-constant and unramified away from $\{0, 1, \infty\}$.

Example

- Curve $X = \mathbb{P}^1$. Smooth, projective ✔
- Map ϕ: given by the univariate polynomial $f = 2x^3 - \frac{3}{2}x + \frac{1}{2}$.
- Ramification ❓ In first chart, values of $t \in \mathbb{C}$ such that $f - t$ has multiple roots.

```
sage: (f - t).discriminant(x).factor()
(-108) * (t - 1) * t
```

Today's application: Belyi maps

Definition

A map $\phi : X \to \mathbb{P}^1$, where

- X is a smooth complex projective algebraic curve,
- ϕ is non-constant and unramified away from $\{0, 1, \infty\}$.

Example

- Curve $X = \mathbb{P}^1$. Smooth, projective ✔
- Map ϕ: given by the univariate polynomial $f = 2x^3 - \frac{3}{2}x + \frac{1}{2}$.
- Ramification ✔ In first chart, values of $t \in \mathbb{C}$ such that $f - t$ has multiple roots.

```
sage: (f - t).discriminant(x).factor()
(-108) * (t - 1) * t
```

Today's application: Belyi maps

Definition

A map $\phi : X \to \mathbb{P}^1$, where

- X is a smooth complex projective algebraic curve,
- ϕ is non-constant and unramified away from $\{0, 1, \infty\}$.

Example

- Curve $X = \mathbb{P}^1$. Smooth, projective ✔
- Map ϕ: given by the univariate polynomial $f = 2x^3 - \frac{3}{2}x + \frac{1}{2}$.
- Ramification ✔ In first chart, values of $t \in \mathbb{C}$ such that $f - t$ has multiple roots.

```
sage: (f - t).discriminant(x).factor()
(-108) * (t - 1) * t
```

This example was taken from the LMFDB (entry 3T2-3_2.1_2.1-a, slightly modified).

Properties

- Belyi maps are uniquely determined by their monodromy for loops around 0 and 1 in $\mathbb{C}$.
- There exists a Belyi map from X if and only X can be defined over $\mathbb{Q}^{\mathrm{al}}$ (Belyi, 1980).

Computing Belyi maps

Properties

- Belyi maps are uniquely determined by their monodromy for loops around 0 and 1 in $\mathbb{C}$.
- There exists a Belyi map from X if and only X can be defined over $\mathbb{Q}^{\mathrm{al}}$ (Belyi, 1980).

Computational goal

Input. $\sigma_0, \sigma_1 \in S_d$. $\langle \sigma_0, \sigma_1 \rangle \subseteq S_d$ transitive subgroup
Output. Equations for X and $\phi : X \to \mathbb{P}^1$ with monodromy (σ_0, σ_1).

See Sijsling and Voight, 2015; Roberts, 2018; Barth and Wenz, 2021.
Klug, Musty, Schiavone, and Voight, 2014 $\rightsquigarrow$ LMFDB Belyi map section.

Computing Belyi maps

Properties

- Belyi maps are uniquely determined by their monodromy for loops around 0 and 1 in $\mathbb{C}$.
- There exists a Belyi map from X if and only X can be defined over $\mathbb{Q}^{\mathrm{al}}$ (Belyi, 1980).

Computational goal

Input. $\sigma_0, \sigma_1 \in S_d$. $\langle \sigma_0, \sigma_1 \rangle \subseteq S_d$ transitive subgroup
Output. Equations for X and $\phi : X \to \mathbb{P}^1$ with monodromy (σ_0, σ_1).

See Sijsling and Voight, 2015; Roberts, 2018; Barth and Wenz, 2021.
Klug, Musty, Schiavone, and Voight, 2014 $\rightsquigarrow$ LMFDB Belyi map section.

Verification step

All known methods require ultimately checking the monodromy of ϕ.

Belyi map verification: previous work

Using symbolic tools

- Elkies, 2013: compute and verify a Belyi map with monodromy group M_{23}.
- Barth and Wenz, 2021: compute and verify Belyi maps of large degree (55–280).

Homotopy continuation type certification

- Schneps, 1994; Bruin, Sijsling, and Zotine, 2019
- Deconinck and van Hoeij, 2001; Bartholdi, Buff, Graf Von Bothmer, and Kröker, 2015

Monodromy in general

- Hauenstein, Haywood, and Liddell, 2014
- Duff and Lee, 2026
- ...

Belyi map section of the LMFDB

- 1111 Belyi maps
- Computed using the method of Klug, Musty, Schiavone, and Voight (2014)
- Contains Belyi maps up to degree 9, contains all Belyi maps with degree ≤ 6
- ! no rigorous monodromy verification

Contribution

- Using Algpath, compute (rigorously) the monodromy of all 1111 Belyi maps entries.
- Along de way, develop Sage scripts to compute the monodromy of Belyi maps in general.
- Results and scripts available at https://gitlab.inria.fr/aguillem/belyi-monodromy.

Number field coefficients

LMFDB data representation

- coefficient field $\mathbb{Q}(\nu)$, given by its minimal polynomial m_ν
- Equations for X and ϕ: coefficients in $\mathbb{Q}(\nu)$ (polynomials in ν)
- Each embedding of $\mathbb{Q}(\nu)$ in $\mathbb{C}$ (a root of m_ν) yields a Belyi map
- For each embedding, the LMFDB provides the monodromy of the associated Belyi map

Entry 5T3-4.1_4.1_2.2.1-a: $m_\nu = T^2 + 1$, $X = \mathbb{P}^1$, $\phi : \frac{1}{4107} \frac{(136\nu+4623)x^5+(15096\nu+513153)x^4}{x^5+(-40\nu+55)x^4+(-13236\nu-12048)x^3+\cdots}$

Number field coefficients

LMFDB data representation

- coefficient field $\mathbb{Q}(\nu)$, given by its minimal polynomial m_ν
- Equations for X and ϕ: coefficients in $\mathbb{Q}(\nu)$ (polynomials in ν)
- Each embedding of $\mathbb{Q}(\nu)$ in $\mathbb{C}$ (a root of m_ν) yields a Belyi map
- For each embedding, the LMFDB provides the monodromy of the associated Belyi map

Entry 5T3-4.1_4.1_2.2.1-a: $m_\nu = T^2 + 1$, $X = \mathbb{P}^1$, $\phi : \frac{1}{4107} \frac{(136\nu+4623)x^5+(15096\nu+513153)x^4}{x^5+(-40\nu+55)x^4+(-13236\nu-12048)x^3+\cdots}$

Problem

The parametrized system F we build to compute the monodromy of a Belyi map may have number field coefficients. How do we handle that in Algpath?

Number field coefficients

Assume a parametrized system F that has coefficients in $\mathbb{Q}(\nu)$ $\leftarrow$ given by m_ν.
Assume an embedding of $\mathbb{Q}(\nu)$ into $\mathbb{C}$, or equivalently α a root of m_ν.

Number field coefficients

Assume a parametrized system F that has coefficients in $\mathbb{Q}(\nu)$ $\leftarrow$ given by m_ν.
Assume an embedding of $\mathbb{Q}(\nu)$ into $\mathbb{C}$, or equivalently α a root of m_ν.

Solution 1

System: $\left\{ F_t(x) = 0 \right.$ Variables: x Starting solution: ζ_0

Number field coefficients

Assume a parametrized system F that has coefficients in $\mathbb{Q}(\nu)$ $\leftarrow$ given by m_ν.
Assume an embedding of $\mathbb{Q}(\nu)$ into $\mathbb{C}$, or equivalently α a root of m_ν.

Solution 1

$$\text{System: } \begin{cases} F_t(x) = 0 \\ m_\nu = 0 \end{cases} \qquad \text{Variables: } x, \nu \qquad \text{Starting solution: } \zeta_0, \alpha$$

Number field coefficients

Assume a parametrized system F that has coefficients in $\mathbb{Q}(\nu)$ $\leftarrow$ given by m_ν.
Assume an embedding of $\mathbb{Q}(\nu)$ into $\mathbb{C}$, or equivalently α a root of m_ν.

Solution 1

$$\text{System: } \begin{cases} F_t(x) = 0 \\ m_\nu = 0 \end{cases} \qquad \text{Variables: } x, \nu \qquad \text{Starting solution: } \zeta_0, \alpha$$

⊕ Termination
⊖ Slows down computations ⚙ rectangular isolating regions

Number field coefficients

Assume a parametrized system F that has coefficients in $\mathbb{Q}(\nu)$ ← given by m_ν.
Assume an embedding of $\mathbb{Q}(\nu)$ into $\mathbb{C}$, or equivalently α a root of m_ν.

Solution 1

$$\text{System: } \begin{cases} F_t(x) = 0 \\ m_\nu = 0 \end{cases} \qquad \text{Variables: } x, \nu \qquad \text{Starting solution: } \zeta_0, \alpha$$

⊕ Termination
⊖ Slows down computations ⚙ rectangular isolating regions

Solution 2 (what we use in practice)

Replace ν by an interval enclosing α $\rightsquigarrow$ F has interval coefficients
⊖ The software may stall if the interval is too wide.
⊕ Faster compared to Solution 1

Practical results

Computation of all triples present in the LMFDB

- whole computation: 17min on 6 threads, using an Intel Core i7-1370P CPU
- 1.2h of CPU time

Found bugs ⚙ being handled

- monodromy may only be correct up to an action of S_3
- bug in the embeddings–monodromy correspondance for 7 entries

Additionally...

Monodromy computations of the map provided by Barth and Wenz (2021).
The two largest maps (degree 266 and 280) are out of reach