

Braid monodromy computations using certified path tracking

Alexandre Guillemot

Joint work with Pierre Lairez

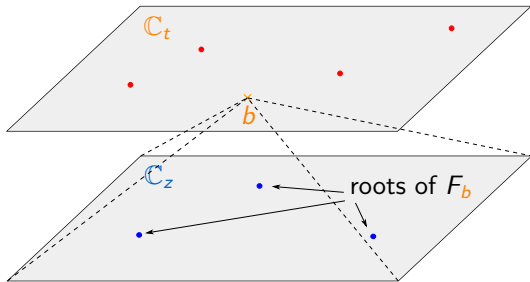
MATHEXP, Inria, France

Séminaire Pascaline

September 25, 2025 | École Normale Supérieure de Lyon



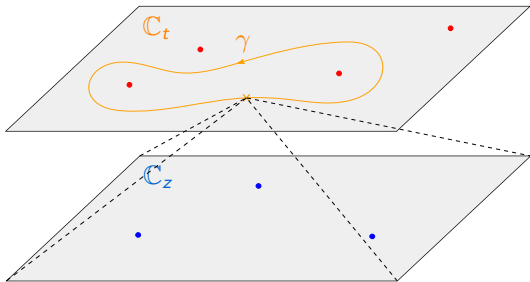
Motivation



Setup

- Let $g \in \mathbb{C}[t, z]$ ($n = \deg_z(g)$),
- define $F_t(z) = g(t, z)$.
- Let $b \in \mathbb{C} \setminus \Sigma$ be a base point,

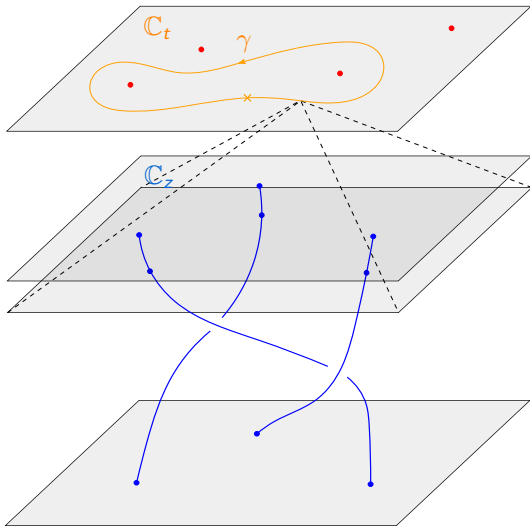
Motivation



Setup

- Let $g \in \mathbb{C}[t, z]$ ($n = \deg_z(g)$),
- define $F_t(z) = g(t, z)$.
- Let $b \in \mathbb{C} \setminus \Sigma$ be a base point,
- let $\gamma : [0, 1] \rightarrow \mathbb{C} \setminus \Sigma$ be a loop starting at b .

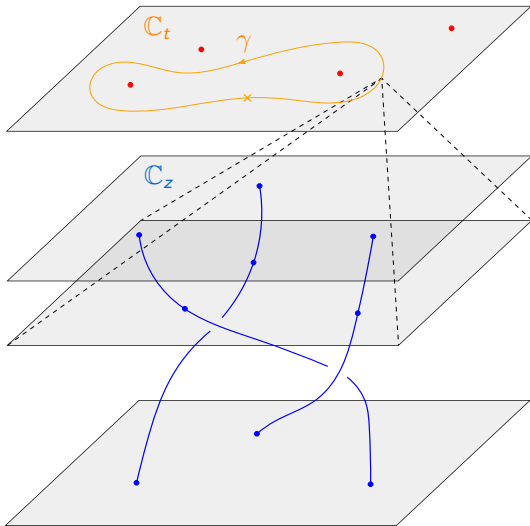
Motivation



Setup

- Let $g \in \mathbb{C}[t, z]$ ($n = \deg_z(g)$),
- define $F_t(z) = g(t, z)$.
- Let $b \in \mathbb{C} \setminus \Sigma$ be a base point,
- let $\gamma : [0, 1] \rightarrow \mathbb{C} \setminus \Sigma$ be a loop starting at b .
- The displacement of all roots of F_t when t moves along γ defines a braid.

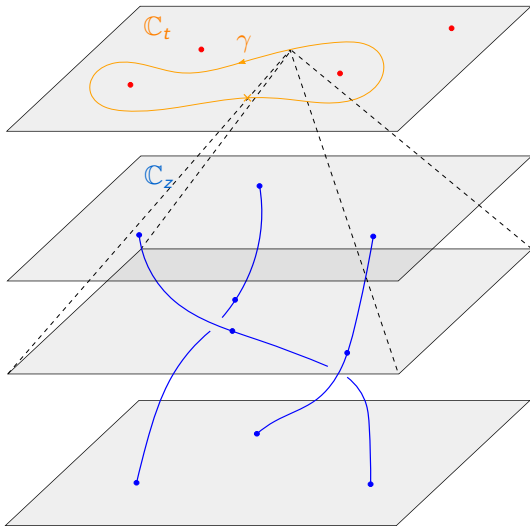
Motivation



Setup

- Let $g \in \mathbb{C}[t, z]$ ($n = \deg_z(g)$),
- define $F_t(z) = g(t, z)$.
- Let $b \in \mathbb{C} \setminus \Sigma$ be a base point,
- let $\gamma : [0, 1] \rightarrow \mathbb{C} \setminus \Sigma$ be a loop starting at b .
- The displacement of all roots of F_t when t moves along γ defines a braid.

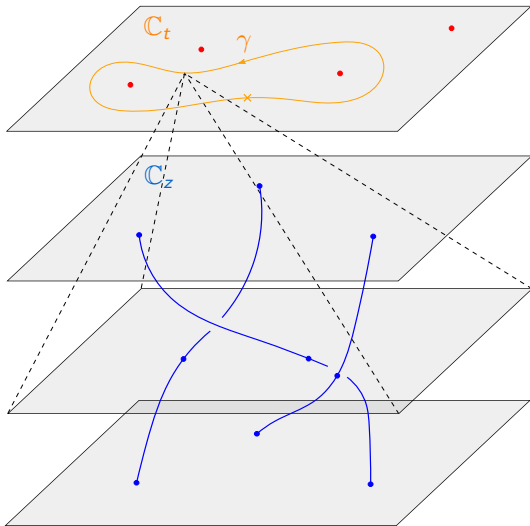
Motivation



Setup

- Let $g \in \mathbb{C}[t, z]$ ($n = \deg_z(g)$),
- define $F_t(z) = g(t, z)$.
- Let $b \in \mathbb{C} \setminus \Sigma$ be a base point,
- let $\gamma : [0, 1] \rightarrow \mathbb{C} \setminus \Sigma$ be a loop starting at b .
- The displacement of all roots of F_t when t moves along γ defines a braid.

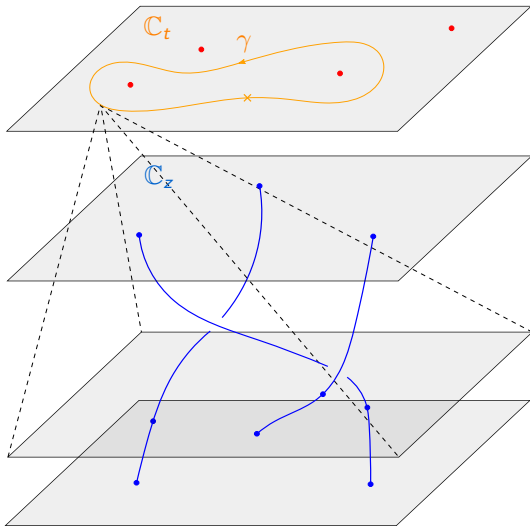
Motivation



Setup

- Let $g \in \mathbb{C}[t, z]$ ($n = \deg_z(g)$),
- define $F_t(z) = g(t, z)$.
- Let $b \in \mathbb{C} \setminus \Sigma$ be a base point,
- let $\gamma : [0, 1] \rightarrow \mathbb{C} \setminus \Sigma$ be a loop starting at b .
- The displacement of all roots of F_t when t moves along γ defines a braid.

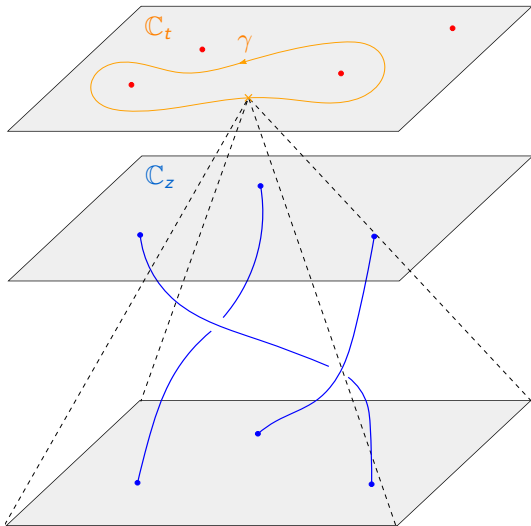
Motivation



Setup

- Let $g \in \mathbb{C}[t, z]$ ($n = \deg_z(g)$),
- define $F_t(z) = g(t, z)$.
- Let $b \in \mathbb{C} \setminus \Sigma$ be a base point,
- let $\gamma : [0, 1] \rightarrow \mathbb{C} \setminus \Sigma$ be a loop starting at b .
- The displacement of all roots of F_t when t moves along γ defines a braid.

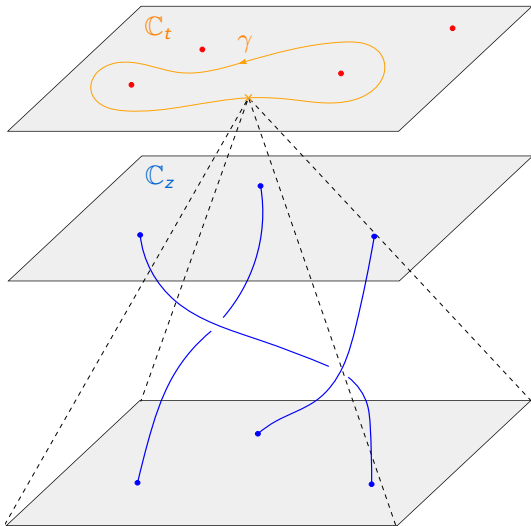
Motivation



Setup

- Let $g \in \mathbb{C}[t, z]$ ($n = \deg_z(g)$),
- define $F_t(z) = g(t, z)$.
- Let $b \in \mathbb{C} \setminus \Sigma$ be a base point,
- let $\gamma : [0, 1] \rightarrow \mathbb{C} \setminus \Sigma$ be a loop starting at b .
- The displacement of all roots of F_t when t moves along γ defines a braid.

Motivation



Setup

- Let $g \in \mathbb{C}[t, z]$ ($n = \deg_z(g)$),
- define $F_t(z) = g(t, z)$.
- Let $b \in \mathbb{C} \setminus \Sigma$ be a base point,
- let $\gamma : [0, 1] \rightarrow \mathbb{C} \setminus \Sigma$ be a loop starting at b .
- The displacement of all roots of F_t when t moves along γ defines a braid.

Algorithmic goal

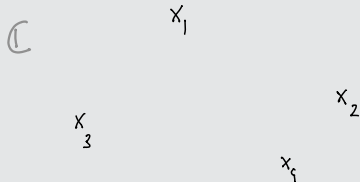
Input: g, γ

Output: the associated braid in terms of Artin's generators

Configurations

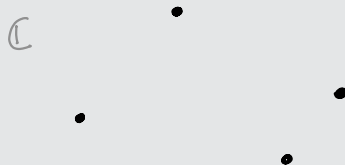
Ordered configurations

$$OC_n = \{(x_1, \dots, x_n) \in \mathbb{C}^n : \forall i \neq j, x_i \neq x_j\}.$$



Configurations

$$C_n = \{\text{subsets of size } n \text{ in } \mathbb{C}\}.$$



"Forget order" projection

$$\begin{array}{ccc} \pi : & OC_n & \rightarrow & C_n \\ & (x_1, \dots, x_n) & \mapsto & \{x_1, \dots, x_n\} \end{array}.$$

Rk: equivalent definition is $C_n = OC_n / \mathfrak{S}_n$.

Braid group

Braid

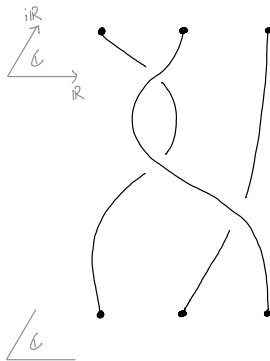
Homotopy class of a path $\beta : [0, 1] \rightarrow C_n$ such that $\beta(0) = \beta(1) = \{1, \dots, n\}$.



Braid group

Braid

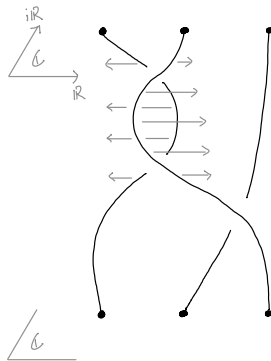
Homotopy class of a path $\beta : [0, 1] \rightarrow C_n$ such that $\beta(0) = \beta(1) = \{1, \dots, n\}$.



Braid group

Braid

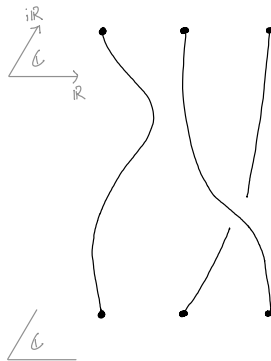
Homotopy class of a path $\beta : [0, 1] \rightarrow C_n$ such that $\beta(0) = \beta(1) = \{1, \dots, n\}$.



Braid group

Braid

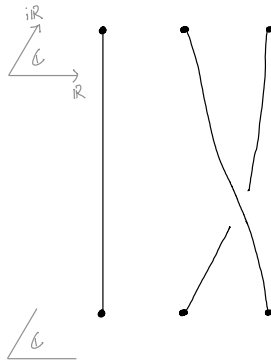
Homotopy class of a path $\beta : [0, 1] \rightarrow C_n$ such that $\beta(0) = \beta(1) = \{1, \dots, n\}$.



Braid group

Braid

Homotopy class of a path $\beta : [0, 1] \rightarrow C_n$ such that $\beta(0) = \beta(1) = \{1, \dots, n\}$.

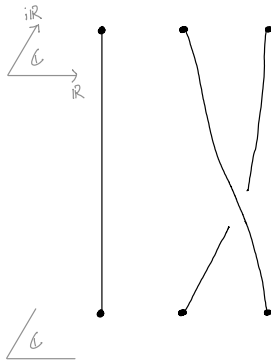


Braid group

Braid

Homotopy class of a path $\beta : [0, 1] \rightarrow C_n$ such that $\beta(0) = \beta(1) = \{1, \dots, n\}$.

In practice, we will manipulate paths in OC_n .



Braid group

Braid

Homotopy class of a path $\beta : [0, 1] \rightarrow C_n$ such that $\beta(0) = \beta(1) = \{1, \dots, n\}$.

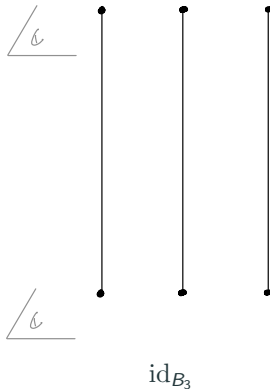
In practice, we will manipulate paths in OC_n .

Braid group B_n

id: class of the constant path equal to $\{1, \dots, n\}$.

Law: $[\beta_1][\beta_2] := [\beta_1 \cdot \beta_2]$

Rk: this is $\pi_1(C_n, \{1, \dots, n\})$.



Braid group

Braid

Homotopy class of a path $\beta : [0, 1] \rightarrow C_n$ such that $\beta(0) = \beta(1) = \{1, \dots, n\}$.

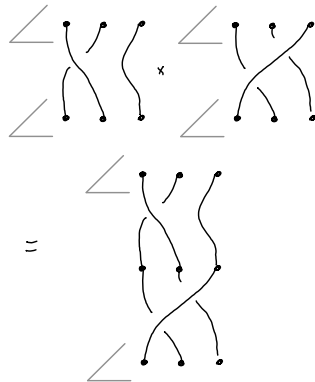
In practice, we will manipulate paths in OC_n .

Braid group B_n

id: class of the constant path equal to $\{1, \dots, n\}$.

Law: $[\beta_1][\beta_2] := [\beta_1 \cdot \beta_2]$

Rk: this is $\pi_1(C_n, \{1, \dots, n\})$.



Braid group

Braid

Homotopy class of a path $\beta : [0, 1] \rightarrow C_n$ such that $\beta(0) = \beta(1) = \{1, \dots, n\}$.

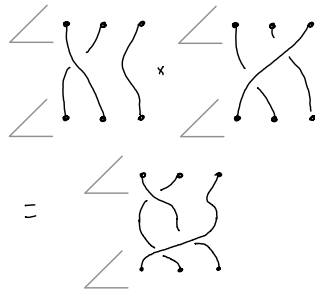
In practice, we will manipulate paths in OC_n .

Braid group B_n

id: class of the constant path equal to $\{1, \dots, n\}$.

Law: $[\beta_1][\beta_2] := [\beta_1 \cdot \beta_2]$

Rk: this is $\pi_1(C_n, \{1, \dots, n\})$.

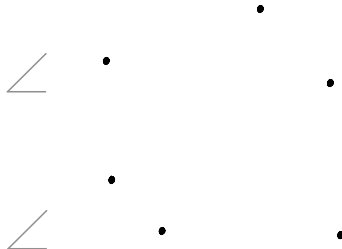


Pseudo braids

Definition

Pseudo braid: homotopy class of a path

$$\beta : [0, 1] \rightarrow C_n.$$

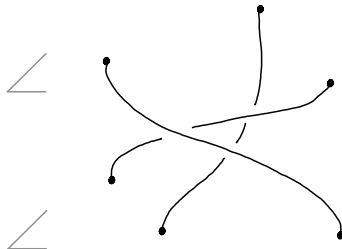


Pseudo braids

Definition

Pseudo braid: homotopy class of a path

$$\beta : [0, 1] \rightarrow C_n.$$



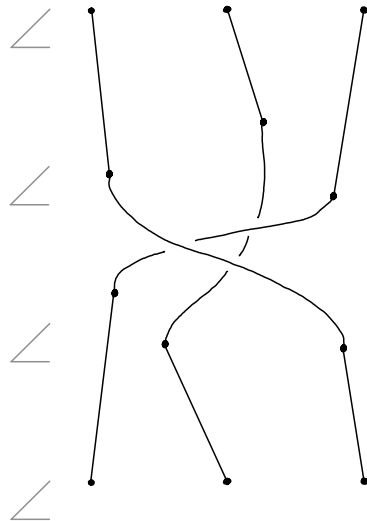
Pseudo braids

Definition

Pseudo braid: homotopy class of a path

$$\beta : [0, 1] \rightarrow C_n.$$

We associate a braid to it by concatenating on top and on bottom specific pseudo-braids to get back to a loop around $\{1, \dots, n\}$.



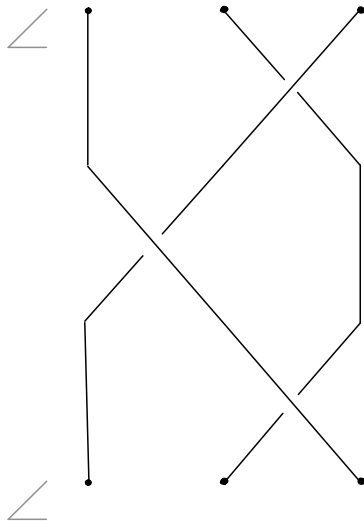
Pseudo braids

Definition

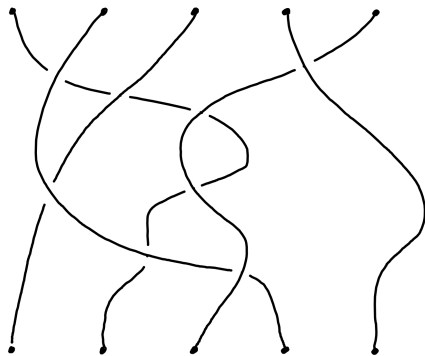
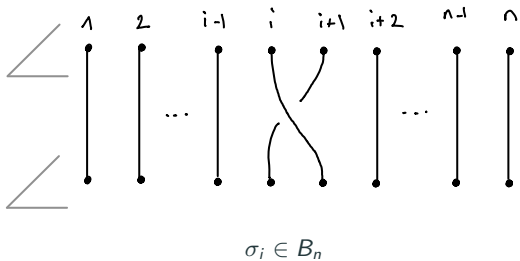
Pseudo braid: homotopy class of a path

$$\beta : [0, 1] \rightarrow C_n.$$

We associate a braid to it by concatenating on top and on bottom specific pseudo-braids to get back to a loop around $\{1, \dots, n\}$.



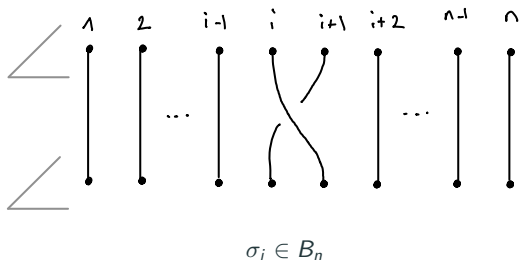
Artin's theorem



Theorem [Artin, 1947]

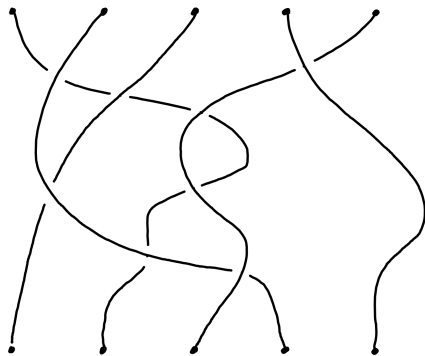
The σ_i 's generate B_n (+ explicit relations).

Artin's theorem



Theorem [Artin, 1947]

The σ_i 's generate B_n (+ explicit relations).



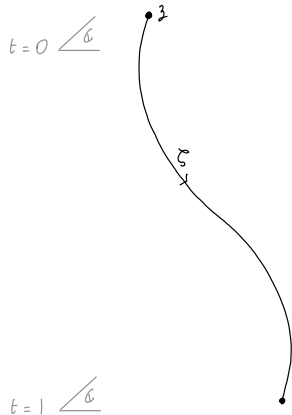
$$\sigma_4 \sigma_1^{-1} \sigma_2^{-1} \sigma_3^{-1} \sigma_3 \sigma_1 \sigma_2 \sigma_3^{-1}$$

Main tool

Certified homotopy continuation

Input: $H : [0, 1] \times \mathbb{C}^r \rightarrow \mathbb{C}^r$ and $z \in \mathbb{C}^r$ such that $H(0, z) = 0$.

There exists $\zeta : [0, 1] \rightarrow \mathbb{C}^r$ such that $H(t, \zeta(t)) = 0$ and $\zeta(0) = z$. Assume it is unique.



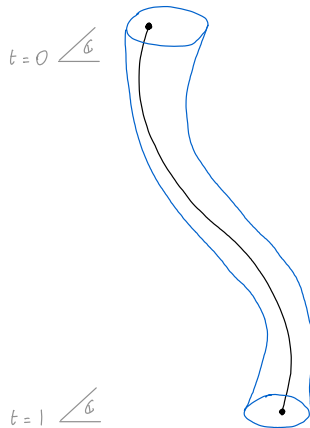
Main tool

Certified homotopy continuation

Input: $H : [0, 1] \times \mathbb{C}^r \rightarrow \mathbb{C}^r$ and $z \in \mathbb{C}^r$ such that $H(0, z) = 0$.

There exists $\zeta : [0, 1] \rightarrow \mathbb{C}^r$ such that $H(t, \zeta(t)) = 0$ and $\zeta(0) = z$. Assume it is unique.

Output: A tubular neighborhood isolating ζ .



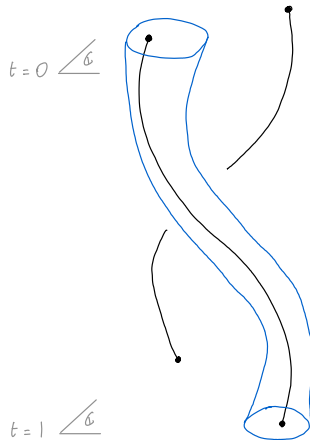
Main tool

Certified homotopy continuation

Input: $H : [0, 1] \times \mathbb{C}^r \rightarrow \mathbb{C}^r$ and $z \in \mathbb{C}^r$ such that $H(0, z) = 0$.

There exists $\zeta : [0, 1] \rightarrow \mathbb{C}^r$ such that $H(t, \zeta(t)) = 0$ and $\zeta(0) = z$. Assume it is unique.

Output: A tubular neighborhood isolating ζ .



Main tool

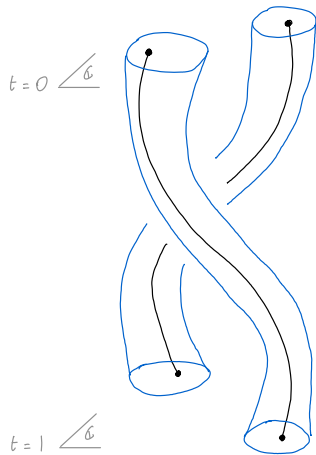
Certified homotopy continuation

Input: $H : [0, 1] \times \mathbb{C}^r \rightarrow \mathbb{C}^r$ and $z \in \mathbb{C}^r$ such that $H(0, z) = 0$.

There exists $\zeta : [0, 1] \rightarrow \mathbb{C}^r$ such that $H(t, \zeta(t)) = 0$ and $\zeta(0) = z$. Assume it is unique.

Output: A tubular neighborhood isolating ζ .

We can do that for every solution at $t = 0$



Main tool

Certified homotopy continuation

Input: $H : [0, 1] \times \mathbb{C}^r \rightarrow \mathbb{C}^r$ and $z \in \mathbb{C}^r$ such that $H(0, z) = 0$.

There exists $\zeta : [0, 1] \rightarrow \mathbb{C}^r$ such that $H(t, \zeta(t)) = 0$ and $\zeta(0) = z$. Assume it is unique.

Output: A tubular neighborhood isolating ζ .

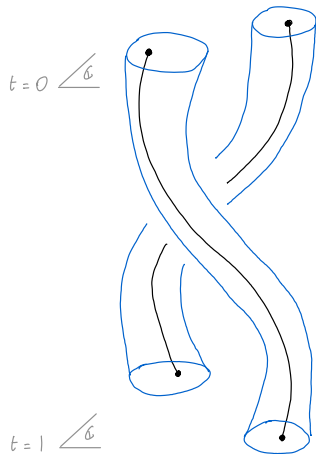
We can do that for every solution at $t = 0$

Application

Recall $g \in \mathbb{C}[t, z]$ and $\gamma : [0, 1] \rightarrow \mathbb{C} \setminus \Sigma$ from first slide.

Apply certified homotopy continuation to

$H(t, z) = g(\gamma(t), z)$.



Certified homotopy continuation

- Kearfott, R. B., & Xing, Z. (1994). An Interval Step Control for Continuation Methods.
- van der Hoeven, J. (2015). Reliable homotopy continuation.
- Xu, J., Burr, M., & Yap, C. (2018). An Approach for Certifying Homotopy Continuation Paths: Univariate Case.
- G., A., & Lairez, P. (2024). Validated Numerics for Algebraic Path Tracking.
- Duff, T., & Lee, K. (2024). Certified homotopy tracking using the Krawczyk method.

Braid computations

- Rodriguez, J. I., & Wang, B. (2017). [Numerical computation of braid groups](#).
- Marco-Buzunariz, M. Á., & Rodríguez, M. (2016). [SIROCCO: a library for certified polynomial root continuation](#).

Today's goal

We now assume $\zeta = (\zeta_1, \dots, \zeta_n) : [0, 1] \rightarrow OC_n$ inducing a loop in C_n i.e. $\pi(\zeta(0)) = \pi(\zeta(1))$.

Goal

Input : ζ (n disjoint tubular neighborhoods around ζ_1, \dots, ζ_n)

Output : A decomposition in standard generators of the braid induced by ζ_1, \dots, ζ_n

Today's goal

We now assume $\zeta = (\zeta_1, \dots, \zeta_n) : [0, 1] \rightarrow OC_n$ inducing a loop in C_n i.e. $\pi(\zeta(0)) = \pi(\zeta(1))$.

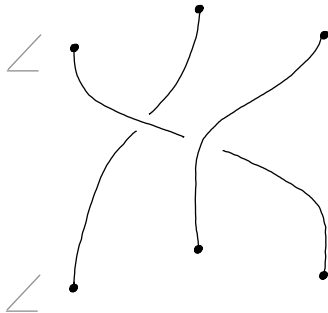
Goal

Input : ζ (n disjoint tubular neighborhoods around ζ_1, \dots, ζ_n)

Output : A decomposition in standard generators of the braid induced by ζ_1, \dots, ζ_n

Overall strategy

! We do not have access to ζ , not even to $\zeta(0)$.



Today's goal

We now assume $\zeta = (\zeta_1, \dots, \zeta_n) : [0, 1] \rightarrow OC_n$ inducing a loop in C_n i.e. $\pi(\zeta(0)) = \pi(\zeta(1))$.

Goal

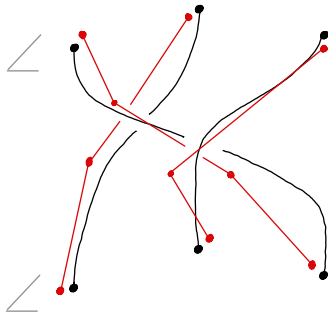
Input : ζ (n disjoint tubular neighborhoods around ζ_1, \dots, ζ_n)

Output : A decomposition in standard generators of the braid induced by ζ_1, \dots, ζ_n

Overall strategy

! We do not have access to ζ , not even to $\zeta(0)$.

1) Find a path $\tilde{\zeta}$ that has same associated braid.



Today's goal

We now assume $\zeta = (\zeta_1, \dots, \zeta_n) : [0, 1] \rightarrow OC_n$ inducing a loop in C_n i.e. $\pi(\zeta(0)) = \pi(\zeta(1))$.

Goal

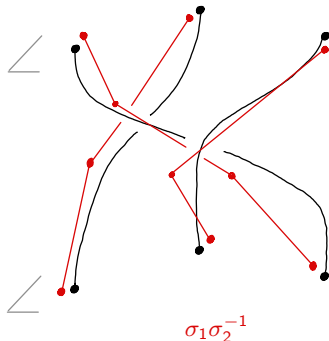
Input : ζ (n disjoint tubular neighborhoods around ζ_1, \dots, ζ_n)

Output : A decomposition in standard generators of the braid induced by ζ_1, \dots, ζ_n

Overall strategy

! We do not have access to ζ , not even to $\zeta(0)$.

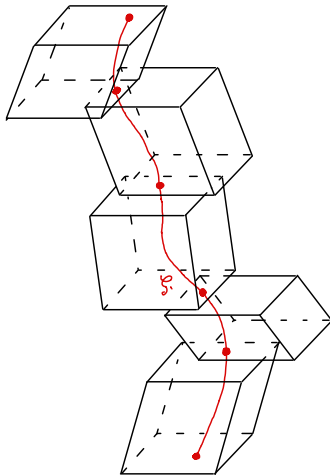
- 1) Find a path $\tilde{\zeta}$ that has same associated braid.
- 2) Decompose $\tilde{\zeta}$.



Algpath vs SIROCCO

SIROCCO [Marco-Buzunariz and Rodríguez, 2016]

- Tubular neighborhoods are piecewise **linear**.
- For each strand ζ_i , computes a piecewise linear path in the tube.
- “Intuitive” (! non generic cases) algorithm on the braid with piecewise linear strands.



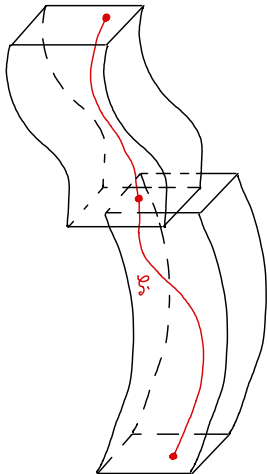
Algpath vs SIROCCO

SIROCCO [Marco-Buzunariz and Rodríguez, 2016]

- Tubular neighborhoods are piecewise **linear**.
- For each strand ζ_i , computes a piecewise linear path in the tube.
- “Intuitive” (! non generic cases) algorithm on the braid with piecewise linear strands.

Algpath [G. and Lairez, 2024]

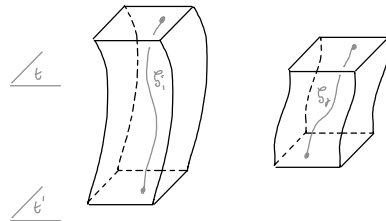
- Tubular neighborhoods are piecewise **cubic**.
- ⊕ Faster than SIROCCO,
- ! Finding a piecewise linear path in the tube requires additional work.



Strand separation

We assume a function $\text{sep}(i, j, t)$ that returns $t' \in (t, 1]$ and a symbol in $\star \in \{\rightarrow, \leftarrow, \rightarrow, \leftarrow\}$, such that for all $s \in [t, t']$,

- $\text{Re}(\zeta_i(s)) < \text{Re}(\zeta_j(s))$ if $\star = \rightarrow$,
- $\text{Re}(\zeta_i(s)) > \text{Re}(\zeta_j(s))$ if $\star = \leftarrow$,
- $\text{Im}(\zeta_i(s)) < \text{Im}(\zeta_j(s))$ if $\star = \rightarrow$,
- $\text{Im}(\zeta_i(s)) > \text{Im}(\zeta_j(s))$ if $\star = \leftarrow$,



$$\text{sep}(i, j, t) = (t', \rightarrow)$$

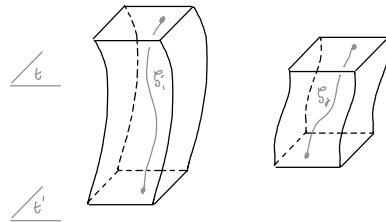
Strand separation

We assume a function $\text{sep}(i, j, t)$ that returns $t' \in (t, 1]$ and a symbol in $\star \in \{\rightarrow, \leftarrow, \rightarrow, \leftarrow\}$, such that for all $s \in [t, t']$,

- $\text{Re}(\zeta_i(s)) < \text{Re}(\zeta_j(s))$ if $\star = \rightarrow$,
- $\text{Re}(\zeta_i(s)) > \text{Re}(\zeta_j(s))$ if $\star = \leftarrow$,
- $\text{Im}(\zeta_i(s)) < \text{Im}(\zeta_j(s))$ if $\star = \rightarrow$,
- $\text{Im}(\zeta_i(s)) > \text{Im}(\zeta_j(s))$ if $\star = \leftarrow$,

Monodromy

We assume a function $\text{monodromy}()$ that returns the monodromy permutation of ζ .



$$\text{sep}(i, j, t) = (t', \rightarrow)$$

$\pi(\zeta(0)) = \pi(\zeta(1)) \Rightarrow \exists \sigma \in \mathfrak{S}_n$ s.t.
for all $i \in [1, n]$, $\zeta_i(1) = \zeta_{\sigma(i)}(0)$.

Recall: $OC_n = \{(x_1, \dots, x_n) \in \mathbb{C}^n : \forall i \neq j, x_i \neq x_j\}$.

Definition

A cell is a pair $c = (R, I)$ of relations on $\{1, \dots, n\}$.

We associate to it a topological space $|c| \subseteq OC_n$ whose points are $(x_1, \dots, x_n) \in OC_n$ such that

- for all $(i, j) \in R$, $\operatorname{Re}(x_i) < \operatorname{Re}(x_j)$,
- for all $(i, j) \in I$, $\operatorname{Im}(x_i) < \operatorname{Im}(x_j)$,

Notation

- $i \xrightarrow{\text{blue}}_c j \iff (i, j) \in R$
- $i \xrightarrow{\text{red}}_c j \iff (i, j) \in I$

Cells

Recall: $OC_n = \{(x_1, \dots, x_n) \in \mathbb{C}^n : \forall i \neq j, x_i \neq x_j\}$.

Definition

A cell is a pair $c = (R, I)$ of relations on $\{1, \dots, n\}$.

We associate to it a topological space $|c| \subseteq OC_n$ whose points are $(x_1, \dots, x_n) \in OC_n$ such that

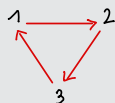
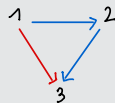
- for all $(i, j) \in R$, $\operatorname{Re}(x_i) < \operatorname{Re}(x_j)$,
- for all $(i, j) \in I$, $\operatorname{Im}(x_i) < \operatorname{Im}(x_j)$,

Notation

- $i \xrightarrow{c} j \iff (i, j) \in R$
- $i \xrightarrow{c} j \iff (i, j) \in I$

Examples

$c = (\emptyset, \emptyset)$: $|c| = OC_n$,



$(1, 2, 3+i) \notin |c|$ $|c| = \emptyset$

Properties of cells

Empty cells

A cell is empty if and only if there is a cycle in R or in I .

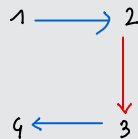
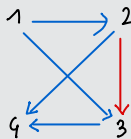
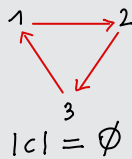
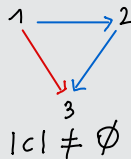
Convex cells

A (non-empty) cell is convex if and only if for all $i, j \in \{1, \dots, n\}$, either $i \rightarrow^* j$ or $j \rightarrow^* i$ or $i \rightarrow^* j$ or $j \rightarrow^* i$. We call this graph property “monochromatic semi-connectedness” (m.s.c. for short).

Intersection of cells

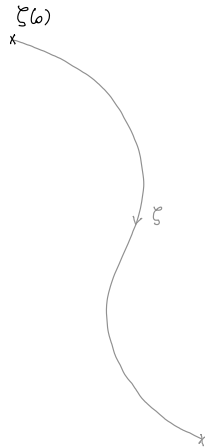
Given $c = (R, I)$ and $c' = (R', I')$ two cells, the space associated to $(R \cup R', I \cup I')$ is $|c| \cap |c'|$.

Examples



Linearization using convex cells

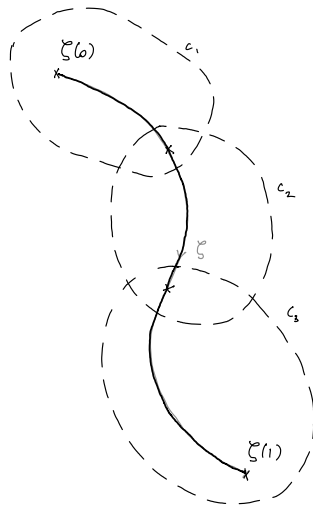
Idea



Linearization using convex cells

Idea

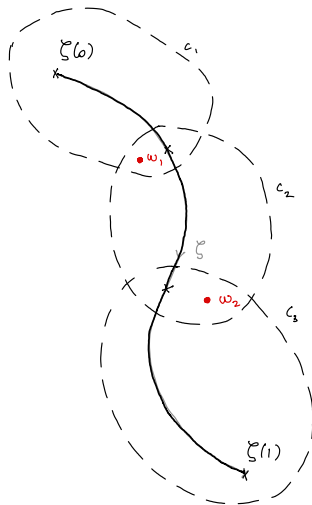
1. Compute a sequence of convex cells covering ζ



Linearization using convex cells

Idea

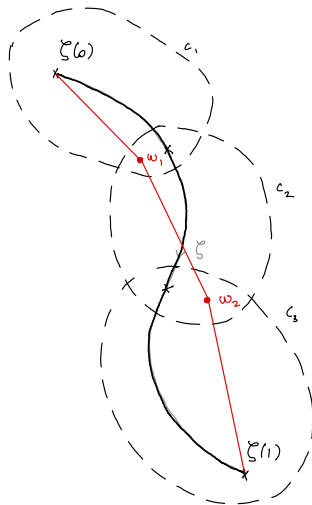
1. Compute a sequence of convex cells covering ζ
2. Find a simplified path covered by the same cells for which the braid is easy to compute



Linearization using convex cells

Idea

1. Compute a sequence of convex cells covering ζ
2. Find a simplified path covered by the same cells for which the braid is easy to compute

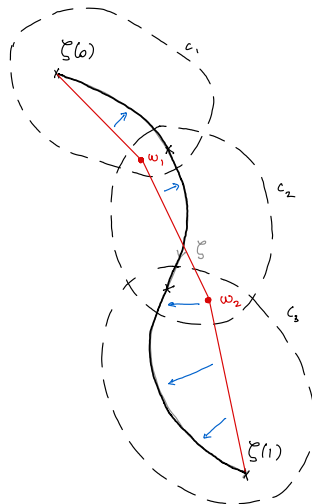


Linearization using convex cells

Idea

1. Compute a sequence of convex cells covering ζ
2. Find a simplified path covered by the same cells for which the braid is easy to compute

- We use sep to compute the sequence of cells
- Correction: convexity of the cells

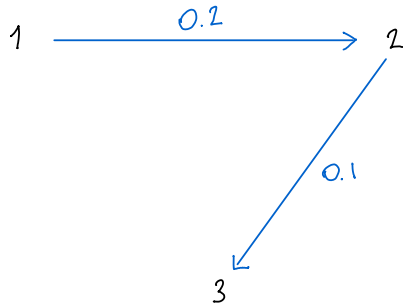


Step 1: compute a sequence of cells

```
def path_to_cells( $\zeta = (\zeta_1, \dots, \zeta_n)$ ):  
  1  $c \leftarrow (1 \xrightarrow{0} 2 \xrightarrow{0} \dots \xrightarrow{0} n, \emptyset)$  # assume that  $\text{Re}(\zeta_1(0)) < \dots < \text{Re}(\zeta_n(0))$   
  2  $res \leftarrow []$   
  3 loop:  
  4    $res.append(c)$   
  5    $i, j, t \leftarrow c.pop()$  # pops the edge with minimal label in  $c$   
  6   if  $t = 1$ : break  
  7    $t', \star \leftarrow \zeta.sep(i, j, t)$  #  $\star \in \{\rightarrow, \leftarrow, \rightarrow, \leftarrow\}$   
  8    $c.insert(i, j, t', \star)$   
  9   Repair monochromatic semi-connectedness # i.e. convexity  
  10  #  $c$  is convex and contains  $\zeta$  on  $[t, s]$  where  $s$  is the smallest time label in  $c$   
  11 return  $res$ 
```

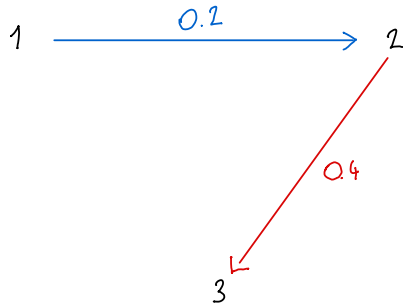
Repair monochromatic semi-connectedness?

```
1 loop:  
2   res.append(c)  
3   i, j, t  $\leftarrow$  c.pop()  
4   if t = 1: break  
5   t', *  $\leftarrow$   $\zeta$ .sep(i, j, t)  
6   c.insert(i, j, t', *)  
7   Repair monochromatic semi-connectedness
```



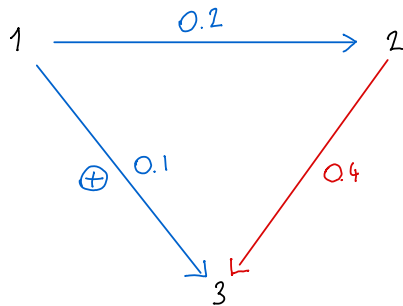
Repair monochromatic semi-connectedness?

```
1 loop:  
2   res.append(c)  
3   i, j, t  $\leftarrow$  c.pop()  
4   if t = 1: break  
5   t', *  $\leftarrow$   $\zeta$ .sep(i, j, t)  
6   c.insert(i, j, t', *)  
7   Repair monochromatic semi-connectedness
```

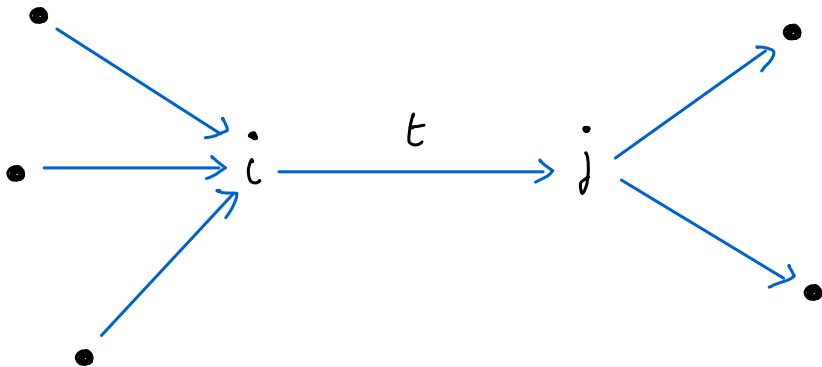


Repair monochromatic semi-connectedness?

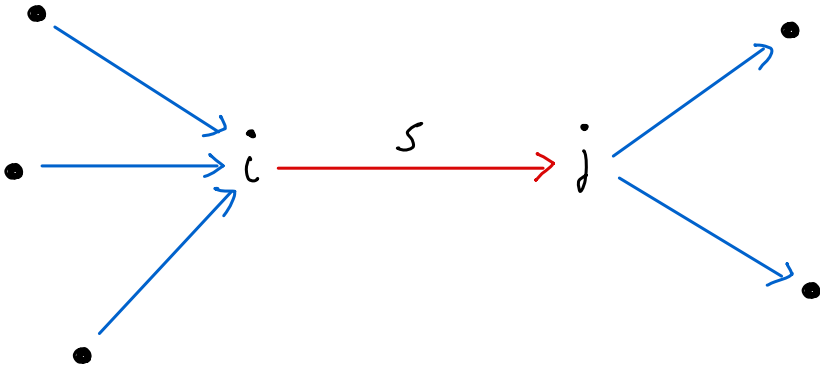
```
1 loop:  
2   res.append(c)  
3   i, j, t  $\leftarrow$  c.pop()  
4   if t = 1: break  
5   t', *  $\leftarrow$   $\zeta$ .sep(i, j, t)  
6   c.insert(i, j, t', *)  
7   Repair monochromatic semi-connectedness
```



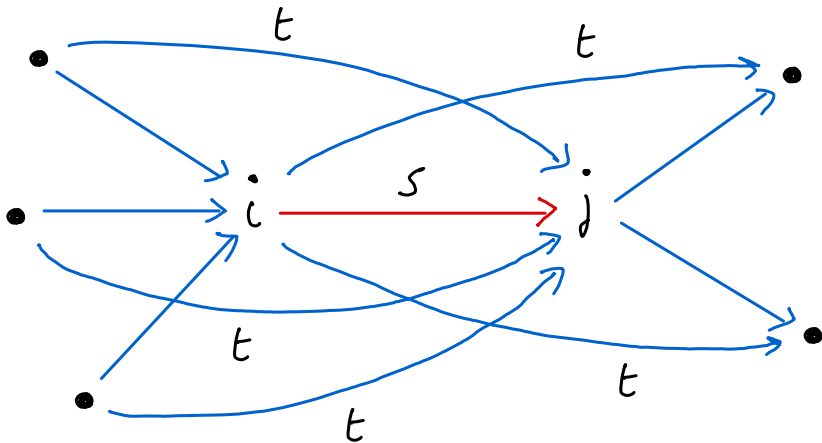
Repair monochromatic semi-connectedness!



Repair monochromatic semi-connectedness!



Repair monochromatic semi-connectedness!



Step 2: linearize ζ

Definition

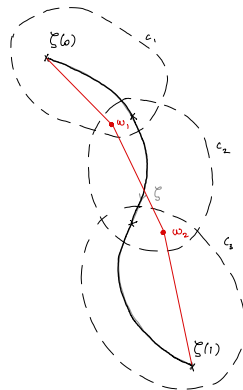
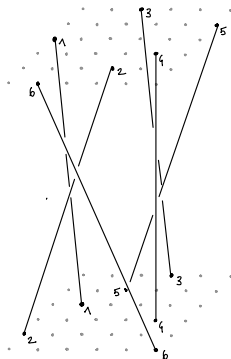
Let $\rho, \iota \in \mathfrak{S}_n$. We define

$$\omega_{\rho, \iota} = (\rho(1) + \mathbf{i}\iota(1), \dots, \rho(n) + \mathbf{i}\iota(n)) \in OC_n$$

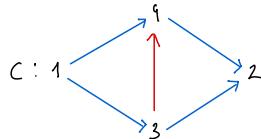
$$\rho = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 3 \end{pmatrix}$$

$$\iota = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 2 & 1 & 3 \end{pmatrix}$$

$$\omega_{\rho, \iota} = \begin{array}{cccc} \cdot & \bullet_1 & \cdot & \cdot \\ \cdot & \cdot & \bullet_4 & \cdot \\ \bullet_2 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \bullet_3 \end{array}$$



How do we find the $\omega_{\rho,\iota}$ in intersecion of cells?



Problem

$c = (R, I)$ nonempty cell. Find ρ, ι such that $\omega_{\rho,\iota} \in |c|$.

Solution

Extend R and I to total orders (“topological sort”).

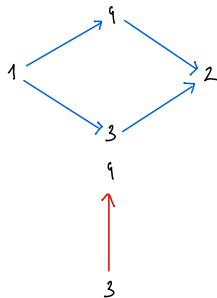
How do we find the $\omega_{\rho,\iota}$ in intersecion of cells?

Problem

$c = (R, I)$ nonempty cell. Find ρ, ι such that $\omega_{\rho,\iota} \in |c|$.

Solution

Extend R and I to total orders (“topological sort”).



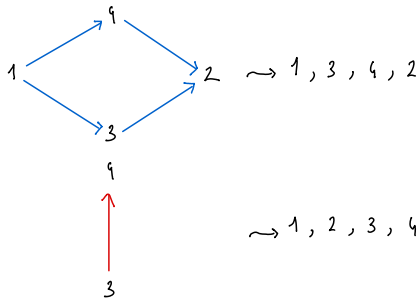
How do we find the $\omega_{\rho,\iota}$ in intersecion of cells?

Problem

$c = (R, I)$ nonempty cell. Find ρ, ι such that $\omega_{\rho,\iota} \in |c|$.

Solution

Extend R and I to total orders (“topological sort”).



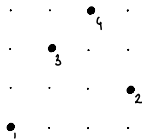
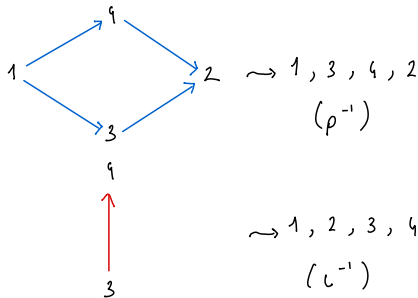
How do we find the $\omega_{\rho, \iota}$ in intersecion of cells?

Problem

$c = (R, I)$ nonempty cell. Find ρ, ι such that $\omega_{\rho, \iota} \in |c|$.

Solution

Extend R and I to total orders (“topological sort”).



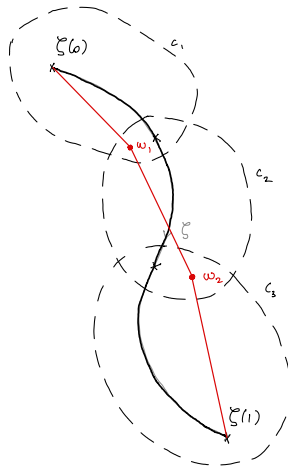
$$= \omega_{\rho, \iota} \in |c|$$

```
def linearize( $\zeta$ )  
1   $cells \leftarrow \zeta.path\_to\_cells()$   
2   $res \leftarrow [(1, 1)]$  #  $\omega_{1,1} \in (1 \xrightarrow{0} 2 \xrightarrow{0} \dots \xrightarrow{0} n, \emptyset)$  the first element of  $cells$   
3  for each pair of successive cells  $c_i, c_{i+1}$  in  $cells$ :  
4      Compute  $\rho, \iota$  such that  $\omega_{\rho, \iota} \in |c_i| \cap |c_{i+1}|$   
5       $res.append((\rho, \iota))$   
6   $\sigma \leftarrow \zeta.monodromy()$  #  $\zeta_i(1) = \zeta_{\sigma(i)}(0)$   
7   $res.append((\sigma, \iota))$   
8  return  $res$ 
```

Main property of linearize

Proposition

The braid associated to ζ and to the linear interpolation of the result of $\zeta.\text{linearize}()$ are equal.



Main property of linearize

...

$res \leftarrow [(1, 1)] \# \omega_{1,1} \in (1 \xrightarrow{0} 2 \xrightarrow{0} \dots \xrightarrow{0} n, \emptyset)$

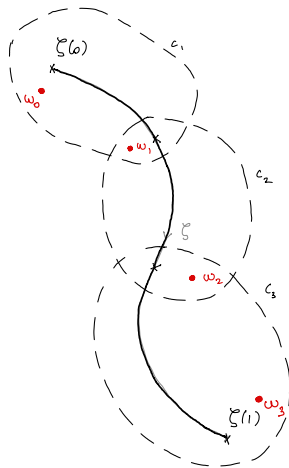
...

$\sigma \leftarrow \zeta.monodromy() \# \zeta_i(1) = \zeta_{\sigma(i)}(0)$

$res.append((\sigma, \iota))$

Proposition

The braid associated to ζ and to the linear interpolation of the result of $\zeta.linearize()$ are equal.



Main property of linearize

...

$res \leftarrow [(1, 1)] \# \omega_{1,1} \in (1 \xrightarrow{0} 2 \xrightarrow{0} \dots \xrightarrow{0} n, \emptyset)$

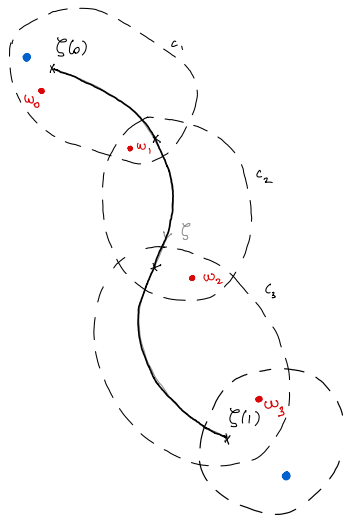
...

$\sigma \leftarrow \zeta.monodromy() \# \zeta_i(1) = \zeta_{\sigma(i)}(0)$

$res.append((\sigma, \iota))$

Proposition

The braid associated to ζ and to the linear interpolation of the result of $\zeta.linearize()$ are equal.



Main property of linearize

...

$res \leftarrow [(1, 1)] \# \omega_{1,1} \in (1 \xrightarrow{0} 2 \xrightarrow{0} \dots \xrightarrow{0} n, \emptyset)$

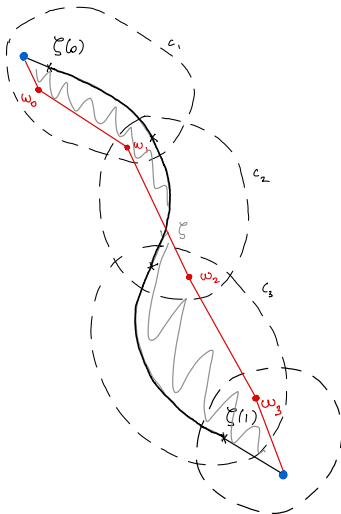
...

$\sigma \leftarrow \zeta.monodromy() \# \zeta_i(1) = \zeta_{\sigma(i)}(0)$

$res.append((\sigma, \iota))$

Proposition

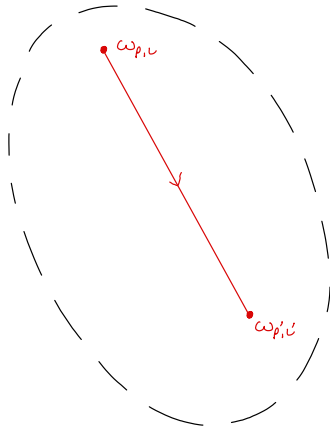
The braid associated to ζ and to the linear interpolation of the result of $\zeta.linearize()$ are equal.



Step 3: decomposition of the linearization in standard generators

Reduction

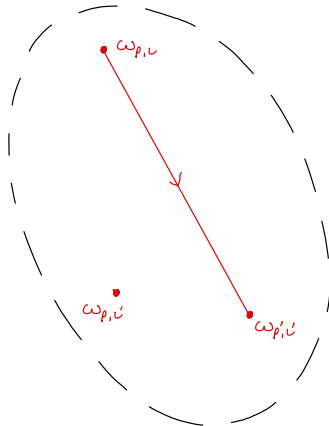
- Computing the braid associated to the whole linearization or to each piece and concatenating the results is equivalent



Step 3: decomposition of the linearization in standard generators

Reduction

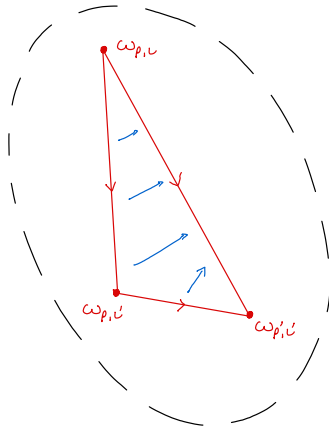
- Computing the braid associated to the whole linearization or to each piece and concatenating the results is equivalent
- Assume $\omega_{\rho,\iota}$ and $\omega_{\rho',\iota'}$ both lie in a m.s.c cell $c = (R, I)$. It means that ρ, ρ' extend R and ι, ι' extend I . **So $\omega_{\rho,\iota'}$ also lies in c !**



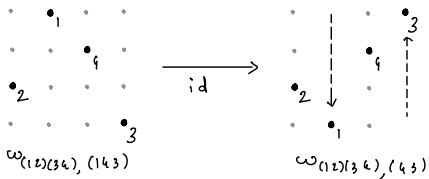
Step 3: decomposition of the linearization in standard generators

Reduction

- Computing the braid associated to the whole linearization or to each piece and concatenating the results is equivalent
- Assume $\omega_{\rho,\iota}$ and $\omega_{\rho',\iota'}$ both lie in a m.s.c cell $c = (R, I)$. It means that ρ, ρ' extend R and ι, ι' extend I . **So $\omega_{\rho,\iota'}$ also lies in c !**
- We compute the braid of $\omega_{\rho,\iota} \rightarrow \omega_{\rho,\iota'}$ then the braid of $\omega_{\rho,\iota'} \rightarrow \omega_{\rho',\iota'}$



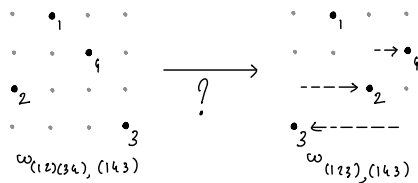
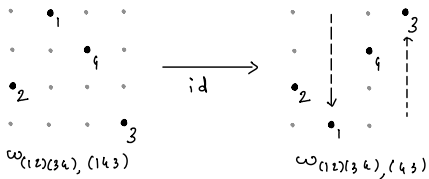
Step 3: decomposition of the linearization in standard generators



$$\omega_{\rho, \iota} \rightarrow \omega_{\rho, \iota'}$$

The induced braid is trivial, as the real part of the strands is constant.

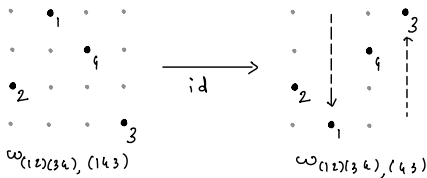
Step 3: decomposition of the linearization in standard generators



$$\omega_{\rho, \iota} \rightarrow \omega_{\rho, \iota'}$$

The induced braid is trivial, as the real part of the strands is constant.

Step 3: decomposition of the linearization in standard generators

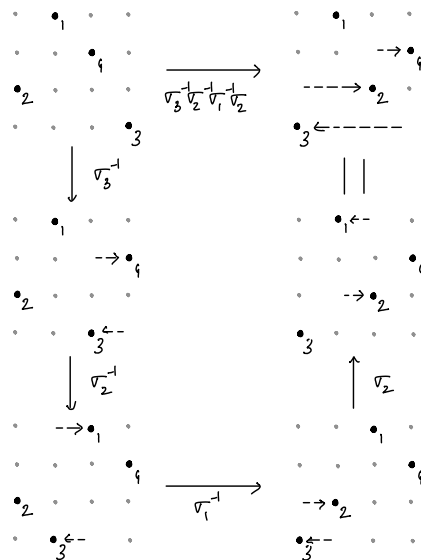


$$\omega_{\rho, \iota} \rightarrow \omega_{\rho, \iota'}$$

The induced braid is trivial, as the real part of the strands is constant.

$$\omega_{\rho, \iota'} \rightarrow \omega_{\rho', \iota'}$$

Let $\rho' \rho^{-1} = s_{i_1} \dots s_{i_r}$ be a decomposition in elementary transpositions. Output $\sigma_{i_1}^{\varepsilon_1} \dots \sigma_{i_r}^{\varepsilon_r}$ with $\varepsilon_1, \dots, \varepsilon_r \in \{\pm 1\}$ computed using ι' .



Optimizations

Cell size

- Worst case, quadratic in the number of strands. But $1 \rightarrow \dots \rightarrow n$ has only $n - 1$ edges.
- In the algorithm presented, we never decrease the number of edges.
- Optimization: before inserting an edge between i and j , check if there is a monochromatic path between i and j and in this case do not insert.

Combine all three steps

- In step 2, we perform multiple topological sorts, but the consecutive cells do not differ by much (an edge deleted and a few inserted)
- Maintain a $\omega_{\rho, \iota}$ and update ρ and ι on cell change.
- Done efficiently using a dynamical topological sort algorithm [Pearce and Kelly, 2007]
- We directly compute the braid of the consecutive $\omega_{\rho, \iota}$.

Conclusion

```
~/2025/code/braid_group cargo run --release
```

```
Finished `release` profile [optimized] target(s) in 0.08s
```

```
Running `target/release/braid_group`
```

```
057011025029047051055061063083030504103103500-10100037073097098027049087097-10590150203-1096-1077092
-1093-1081092024072-1084018-1026094-1095-1010088087-1017-1024-1016-1025024082015-1063-1086-1013014-1
013-106-1081-1065087-1012-1073-1011-1088-1096-10930940405-1027-1010-1066062071074-106-1070604-101509
-1093023016-1092-108-1020-107-1075-106-105-104-1091022021089-1020073085090-1022091-103-1092-106093-1
082-1083-102-1082094-1095-1012-106700-101-100-1096-1035-1068076-1077-1097-1014015-1014-1098-10600940
92-1091078-1013-1014070-1069070059-1021083079-1080-1092071015023-1017-109-1018010019081-1018093092-1
017084083-1082-1083-1084-1016072079-1012-1076-1013085-1073086-1036074081087-1088-1015089-10140870130
12017-1018028090-1091-1078092-1093-1094-1095-1098097092-1096-1097-1024098-1094075029-1015076088087-1
033095094093092011077078068067-109109007908001009019-1079-108407-10807030078060805091031-10890408108
2047-103088020021-1083081084085076-1032033-10860201094087077-1075076088089075-1090091066065-1022-108
5-1084092093094095023096097024082-1083082019020-1034-1035036081-1074-1073072025063064063-1073-107402
6-1037-1038-1039062061060-1059-1060061-1023-1012-1013058057-1040041-1022014015-1042-1043056055-1054-
1053044-1072-1073045026025-1084-1074-1075086076-1077052051-1038085050049-1024-1023046-1047078-107909
800084048078082-1080030-1034052-1074076090-1023-1024036-1079078-104-1021046047070-1046-1028-1068-108
609206023054066-1088-1094-102032-1049050-1038-1010026-1056-1058-1072040-1044051-1085060-1052012096-1
045044-1024023-1017053054-1023024-1077076-1086075084-1042043042-1055-1056062-1074-1025-1026073072-10
57-1058038060061060-1059-1022049-1041-1014-1015-1014023-1060061-1040013012-1085-1062039038-1074073-1
037-1060-1063064-1036035034-1081-1065-1066072026-1025082083024025-1026082-1084-1073074-1032033032-10
33-1023022-1019-1020-1019097096031-1095094093092085030067-106808-1021-1020091090075-1076075087077-10
88076-1089088087086085081047-1098-1097-1029-1028096-1084094095-1094-1093-1083082081091092-1091-1090-
1078088089-1088-1079-1087-1082-1086-10102059-1060077018017-1085-1084-1083-1082-1078-1030408808089-1
018-1019-1018019069-1079084016-1078096077070081-10506080-1070800-101-1094-1079-1075-1076075078-1017-
101502-103-1011010-1091010077-1090-1076-101101604-105-1022-106-107-1074073014013-1014-1015-101408-107
5-1074-109-1010-1023-1072021026-1012013071022073-1067070069070025089-1072-1014015093-1092-1094093092
071-1016070-1017088-1095096-1011-1012-1015-1027-1013-1014-1015-1016-1094095-1018019086015-1026068067
011-1085069-1013014068-1087-1033017-1086-1018-1082-108304-105-106-1094-1081-108204050607091093-10100
92-1088021-1089-1023020021022021020087088-1023019-106-1020-1021-1022-105-104093-1020019-1020-1094065
013095096-1020302-10100-101-103-107-109-1013-1017-1019-1021-1023-1033-1035-1039-1043-1045-1053-1063-
1065-1067-1069-1071-1075-1079-1085-1087-1089-1091-1095-1097-1098
```

