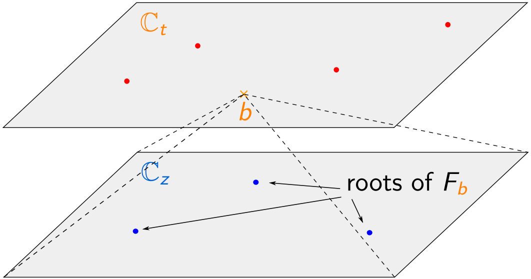
Braid monodromy computations using certified path tracking

Alexandre Guillemot
Joint work with Pierre Lairez
MATHEXP, Inria, France

Séminaire Pascaline
September 25, 2025 | École Normale Supérieure de Lyon

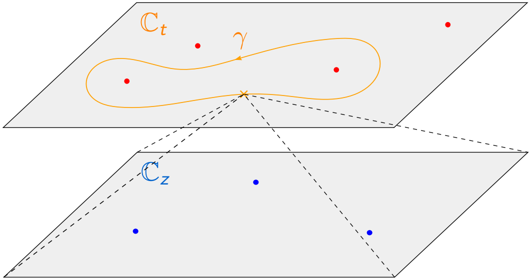
Motivation



Setup

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

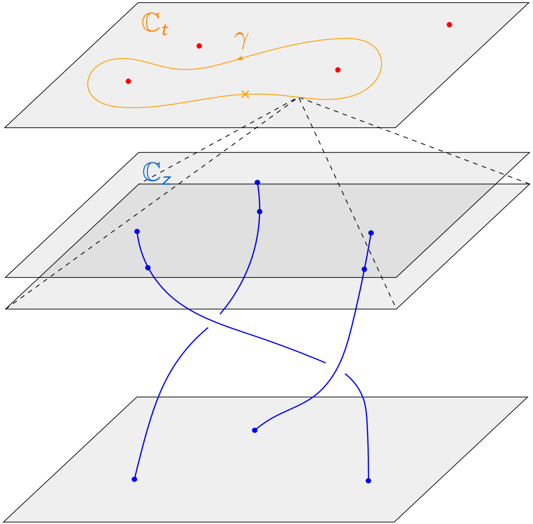
Motivation



Setup

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

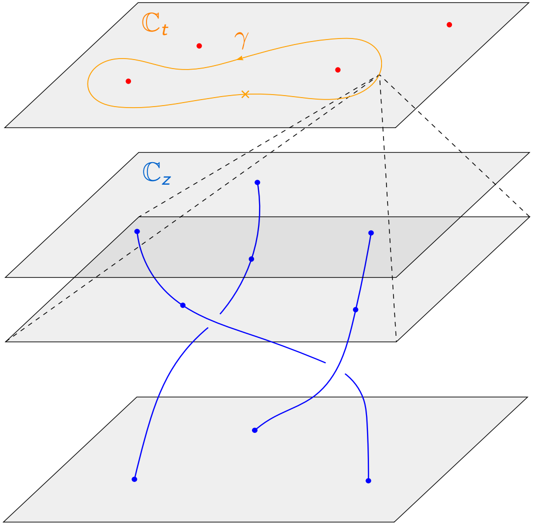
Motivation



Setup

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

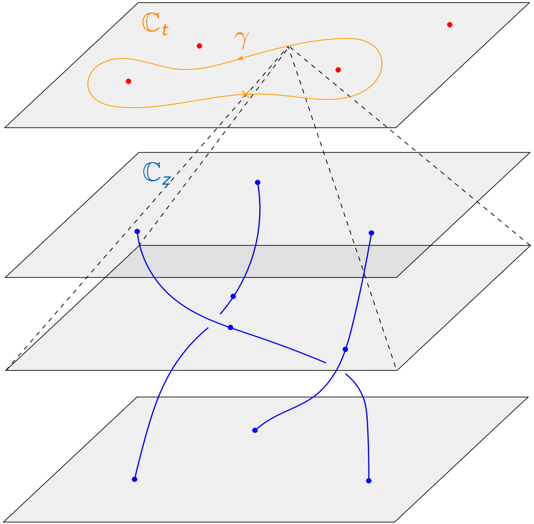
Motivation



Setup

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

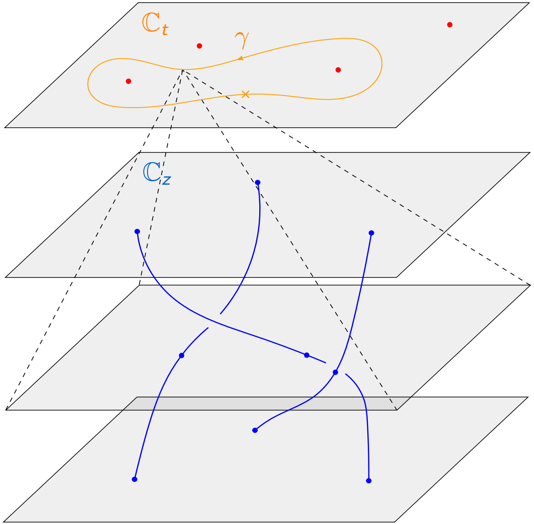
Motivation



Setup

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

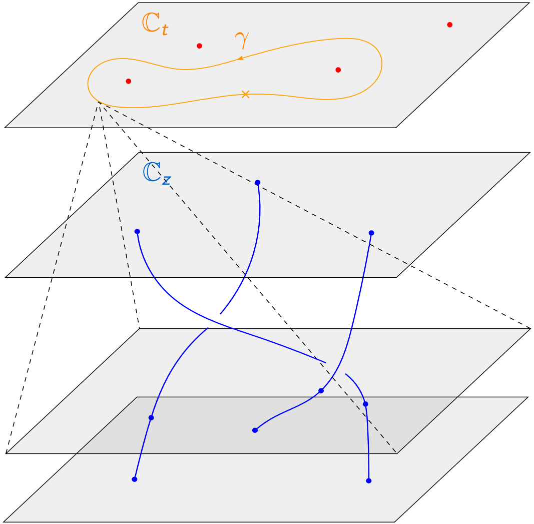
Motivation



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 $\gamma : [0,1] \to \mathbb{C}\setminus\Sigma$ be a loop starting at b.
- The displacement of all roots of F_t when t moves along γ defines a braid.

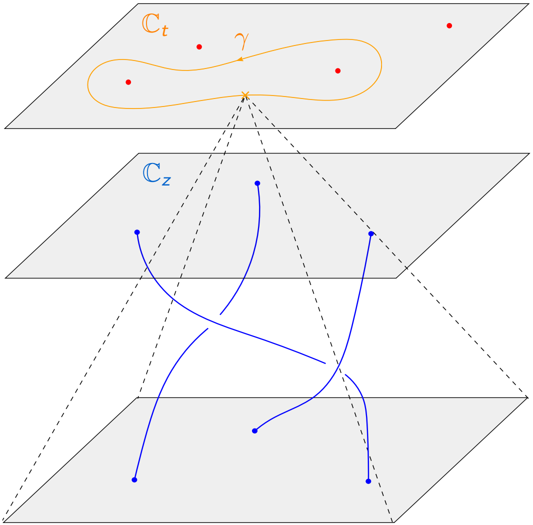
Motivation



Setup

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

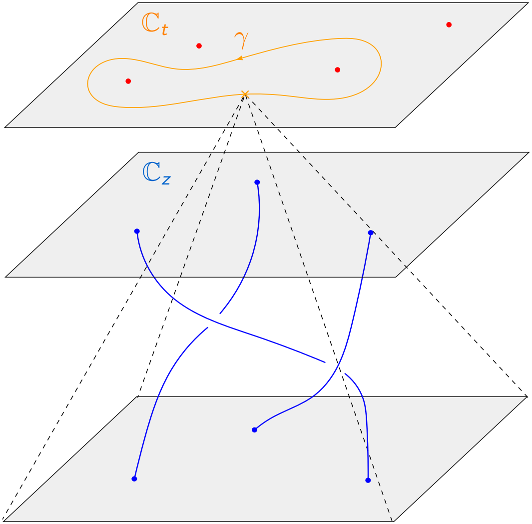
Motivation



Setup

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

Algorithmic goal

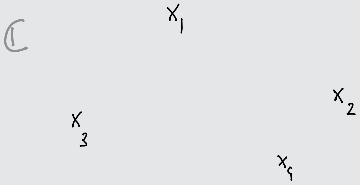
Input: g, γ

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

Configurations

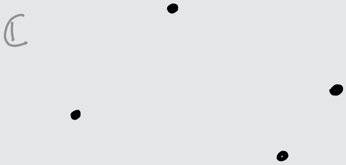
Ordered configurations

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



Configurations

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



"Forget order" projection

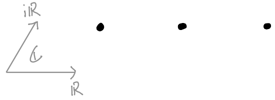
$$\pi : \begin{array}{ccc} OC_n & \to & C_n \\ (x_1, \dots, x_n) & \mapsto & \{x_1, \dots, x_n\} \end{array}.$$

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

Braid group

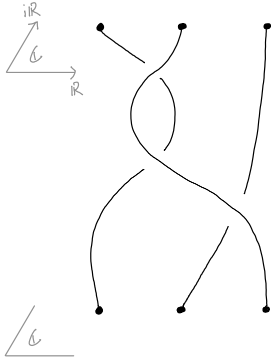
Braid

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



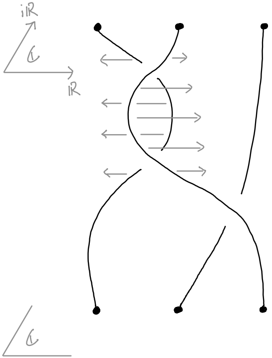
Braid

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



Braid

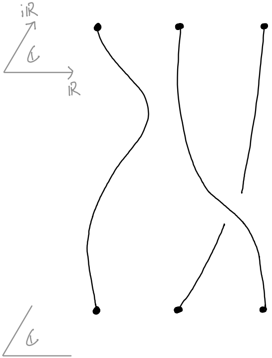
Homotopy class of a path $\beta : [0,1] \to C_n$ such that $\beta(0) = \beta(1) = \{1, \dots, n\}$.



Braid group

Braid

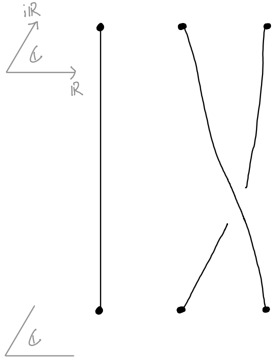
Homotopy class of a path $\beta : [0,1] \to C_n$ such that $\beta(0) = \beta(1) = \{1, \ldots, n\}$.



Braid group

Braid

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

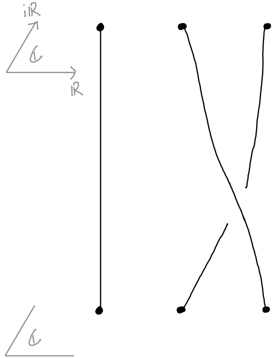


Braid group

Braid

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

In practice, we will manipulate paths in OC_n.



Braid group

Braid

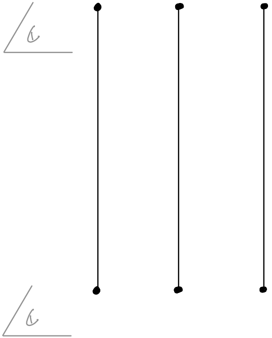
Homotopy class of a path $\beta : [0,1] \to C_n$ such that $\beta(0) = \beta(1) = \{1, \ldots, n\}$.

In practice, we will manipulate paths in OC_n.

Braid group B_n

id: class of the constant path equal to $\{1, \ldots, n\}$.
Law: $[\beta_1][\beta_2] := [\beta_1 \cdot \beta_2]$

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



id_{B_3}

Braid group

Braid

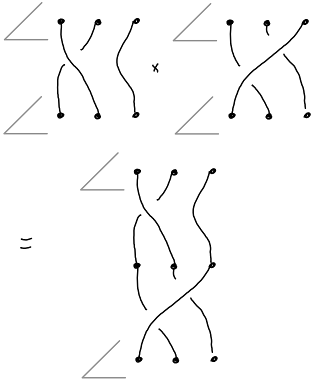
Homotopy class of a path $\beta : [0,1] \to C_n$ such that $\beta(0) = \beta(1) = \{1, \dots, n\}$.

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

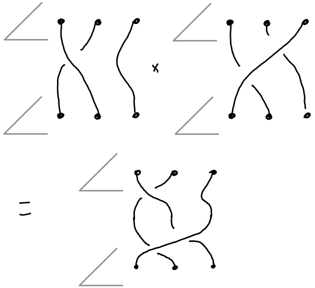
Homotopy class of a path $\beta : [0,1] \to C_n$ such that $\beta(0) = \beta(1) = \{1, \dots, n\}$.

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

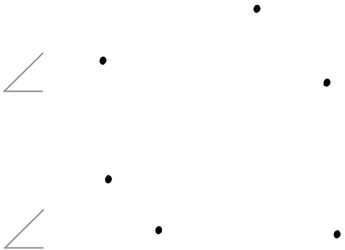
Rk: this is $\pi_1(C_n, \{1, \dots, n\})$.



Pseudo braids

Definition

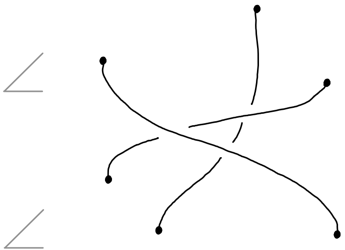
Pseudo braid: homotopy class of a path $\beta : [0,1] \to C_n$.



Pseudo braids

Definition

Pseudo braid: homotopy class of a path $\beta : [0,1] \to C_n$.

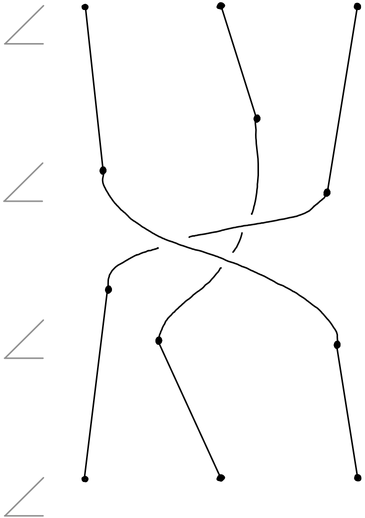


Pseudo braids

Definition

Pseudo braid: homotopy class of a path $\beta : [0,1] \to C_n$.

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

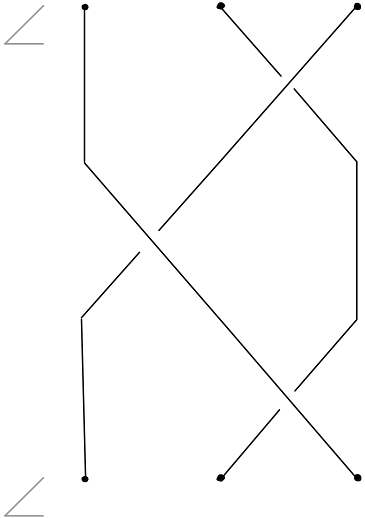


Pseudo braids

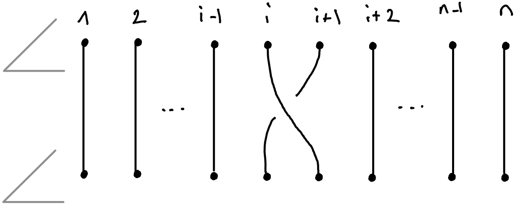
Definition

Pseudo braid: homotopy class of a path $\beta : [0, 1] \to C_n$.

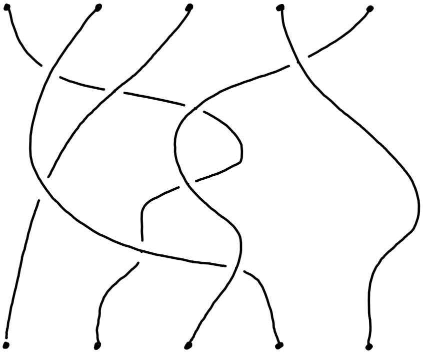
We associate a braid to it by concatenating on top and on bottom specific pseudo-braids to get back to a loop around $\{1, \ldots, n\}$.



Artin's theorem



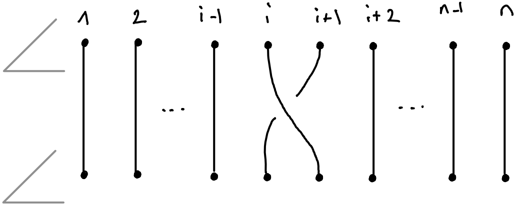
$\sigma_i \in B_n$



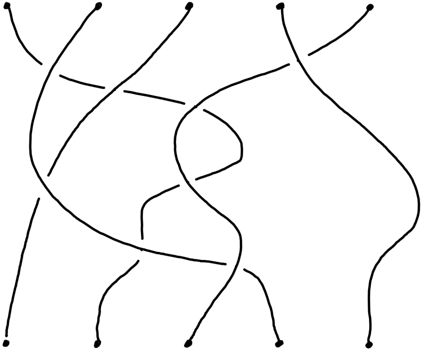
Theorem [Artin, 1947]

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

Artin's theorem



$\sigma_i \in B_n$



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

Theorem [Artin, 1947]

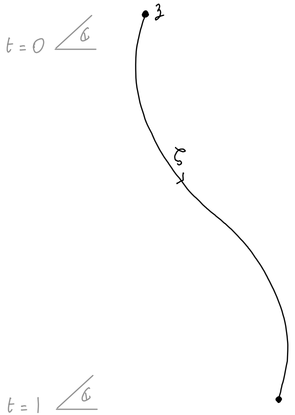
The σ_i's generate B_n (+ explicit relations).

Main tool

Certified homotopy continuation

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

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



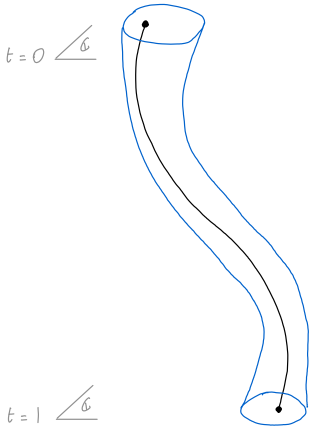
Main tool

Certified homotopy continuation

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

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

Output: A tubular neighborhood isolating ζ.



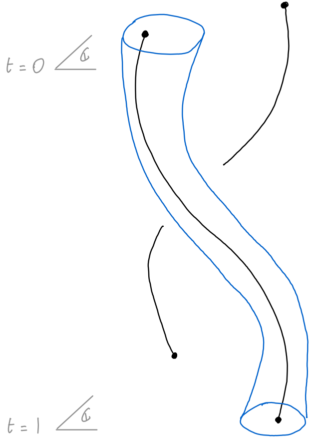
Main tool

Certified homotopy continuation

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

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

Output: A tubular neighborhood isolating ζ.



Main tool

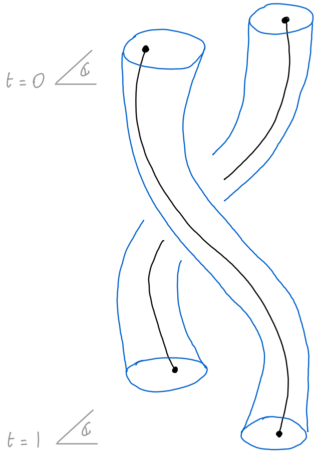
Certified homotopy continuation

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

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

Output: A tubular neighborhood isolating ζ.

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



Main tool

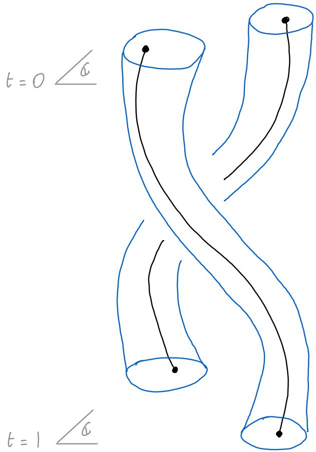
Certified homotopy continuation

Input: $H : [0,1] \times \mathbb{C}^r \to \mathbb{C}^r$ and $z \in \mathbb{C}^r$ such that $H(0, z) = 0$.
There exists $\zeta : [0,1] \to \mathbb{C}^r$ *such that* $H(t, \zeta(t)) = 0$ *and* $\zeta(0) = z$. *Assume it is unique.*
Output: A tubular neighborhood isolating ζ.

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

Application

Recall $g \in \mathbb{C}[t, z]$ and $\gamma : [0,1] \to \mathbb{C} \backslash \Sigma$ from first slide.
Apply certified homotopy continuation to $H(t, z) = g(\gamma(t), z)$.



Related work

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 $\zeta = (\zeta_1, \ldots, \zeta_n) : [0,1] \to OC_n$ inducing a loop in C_n i.e. $\pi(\zeta(0)) = \pi(\zeta(1))$.

Goal

Input : ζ (n disjoint tubular neighborhoods around $\zeta_1, \ldots, \zeta_n$)

Output : A decomposition in standard generators of the braid induced by $\zeta_1, \ldots, \zeta_n$

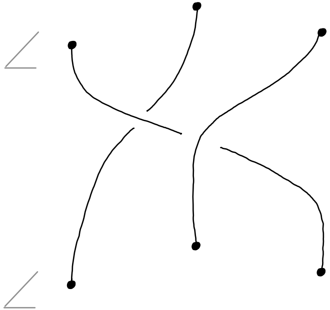
Today's goal

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

Goal

Input : ζ (n disjoint tubular neighborhoods around $\zeta_1, \ldots, \zeta_n$)

Output : A decomposition in standard generators of the braid induced by $\zeta_1, \ldots, \zeta_n$



Overall strategy

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

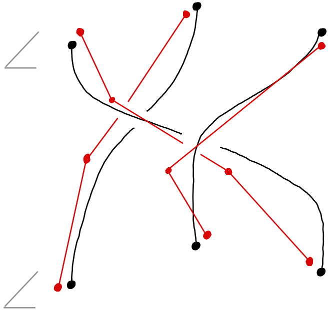
Today's goal

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

Goal

Input : ζ (n disjoint tubular neighborhoods around $\zeta_1, \dots, \zeta_n$)

Output : A decomposition in standard generators of the braid induced by $\zeta_1, \dots, \zeta_n$



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 $\zeta = (\zeta_1, \ldots, \zeta_n) : [0,1] \to OC_n$ inducing a loop in C_n i.e. $\pi(\zeta(0)) = \pi(\zeta(1))$.

Goal

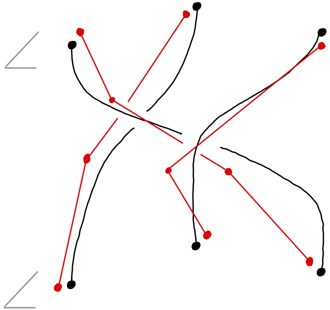
Input : ζ (n disjoint tubular neighborhoods around $\zeta_1, \ldots, \zeta_n$)

Output : A decomposition in standard generators of the braid induced by $\zeta_1, \ldots, \zeta_n$

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 $\tilde{\zeta}$.

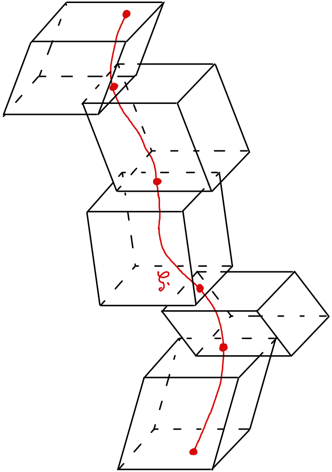


$\sigma_1 \sigma_2^{-1}$

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.



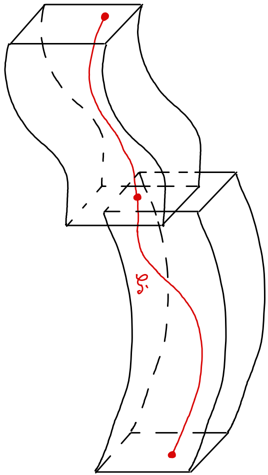
Algpath vs SIROCCO

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]

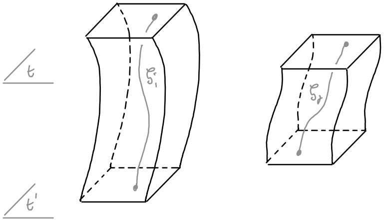
- Tubular neighborhoods are piecewise **cubic**.
- ➕ Faster than SIROCCO,
- **!** Finding a piecewise linear path in the tube requires additional work.



Strand separation

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

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



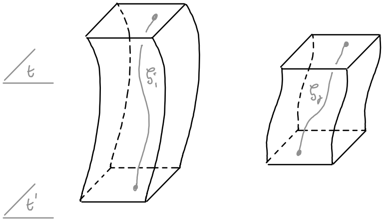
$\mathtt{sep}(i, j, t) = (t', {\color{blue}\rightarrow})$

Interface

Strand separation

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

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



$\mathtt{sep}(i, j, t) = (t', {\color{blue}\rightarrow})$

Monodromy

We assume a fonction `monodromy()` that returns the monodromy permutation of ζ.

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

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

Notation

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

Cells

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

Definition

A cell is a pair $c = (R, I)$ of relations on $\{1, \ldots, n\}$.
We associate to it a topological space $|c| \subseteq OC_n$ whose points are $(x_1, \ldots, x_n) \in OC_n$ such that

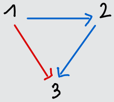
- for all $(i,j) \in R$, $\mathrm{Re}(x_i) < \mathrm{Re}(x_j)$,
- for all $(i,j) \in I$, $\mathrm{Im}(x_i) < \mathrm{Im}(x_j)$,

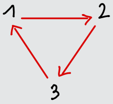
Notation

- $i \to_c j \iff (i,j) \in R$
- $i \to_c j \iff (i,j) \in I$

Examples

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





$(1, 2, 3+i) \in |c|$ $\quad |c| = \varnothing$

Properties of cells

Empty cells

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

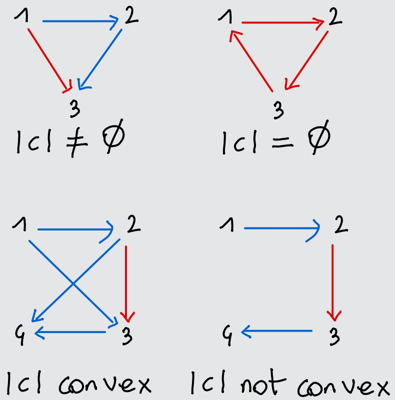
Convex cells

A (non-empty) cell is convex if and only if for all $i, j \in \{1, \ldots, n\}$, 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

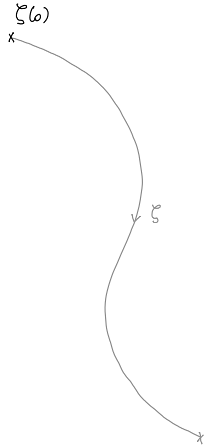
Given $c = (R, I)$ and $c' = (R', I')$ two cells, the space associated to $(R \cup R', I \cup I')$ is $|c| \cap |c'|$.

Examples



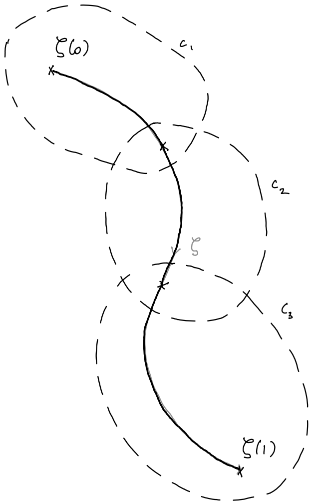
Linearization using convex cells

Idea



Idea

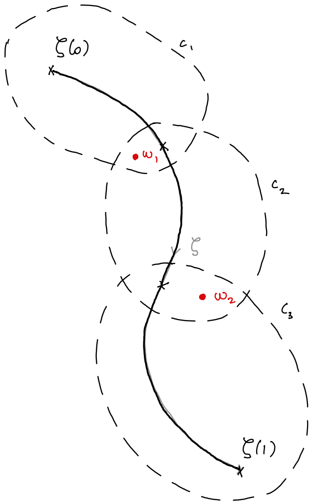
1. Compute a sequence of convex cells covering ζ



Linearization using convex cells

Idea

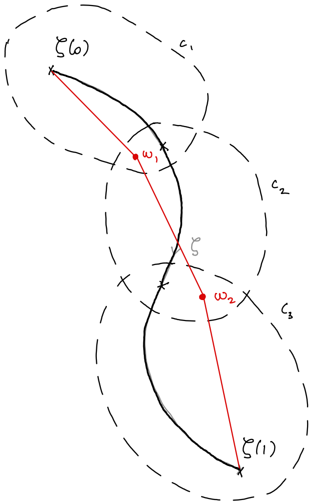
1. Compute a sequence of convex cells covering ζ
2. Find a simplified path covered by the same cells for which the braid is easy to compute



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

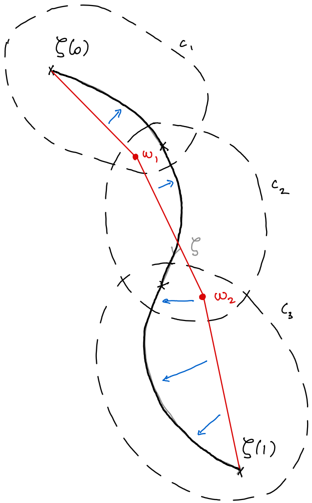


Linearization using convex cells

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 $\texttt{path\_to\_cells}(\zeta = (\zeta_1, \dots, \zeta_n))$**:**

$c \leftarrow (1 \xrightarrow{0} 2 \xrightarrow{0} \dots \xrightarrow{0} n, \varnothing)$ # assume that $\mathrm{Re}(\zeta_1(0)) < \dots < \mathrm{Re}(\zeta_n(0))$

$res \leftarrow []$

loop:

  $res.\mathrm{append}(c)$

  $i, j, t \leftarrow c.\mathrm{pop}()$ # pops the edge with minimal label in c

  if $t = 1$: **break**

  $t', \star \leftarrow \zeta.\mathrm{sep}(i, j, t)$ # $\star \in \{\rightarrow, \leftarrow, \rightarrow, \leftarrow\}$

  $c.\texttt{insert}(i, j, t', \star)$

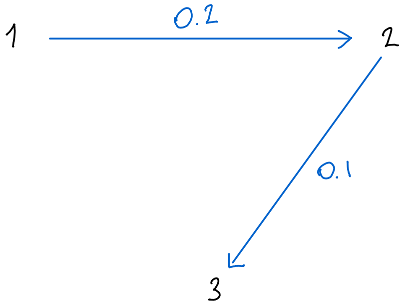
  Repair monochromatic semi-connectedness # i.e. convexity

  # c is convex and contains ζ on $[t, s]$ where s is the smallest time label in c

return res

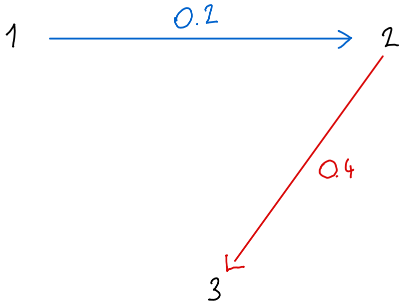
Repair monochromatic semi-connectedness?

```
loop:
    res.append(c)
    i, j, t ← c.pop()
    if t = 1: break
    t′, ⋆ ← ζ.sep(i, j, t)
    c.insert(i, j, t′, ⋆)
    Repair monochromatic semi-connectedness
```



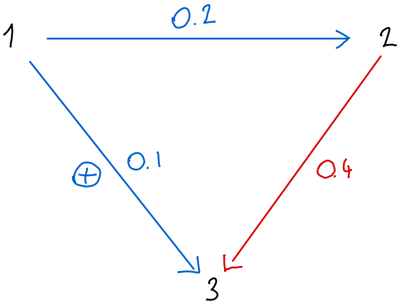
Repair monochromatic semi-connectedness?

loop:
  res.append(c)
  $i, j, t \leftarrow c$.pop()
  if $t = 1$: **break**
  $t', \star \leftarrow \zeta$.sep($i, j, t$)
  c.insert($i, j, t', \star$)
  Repair monochromatic semi-connectedness

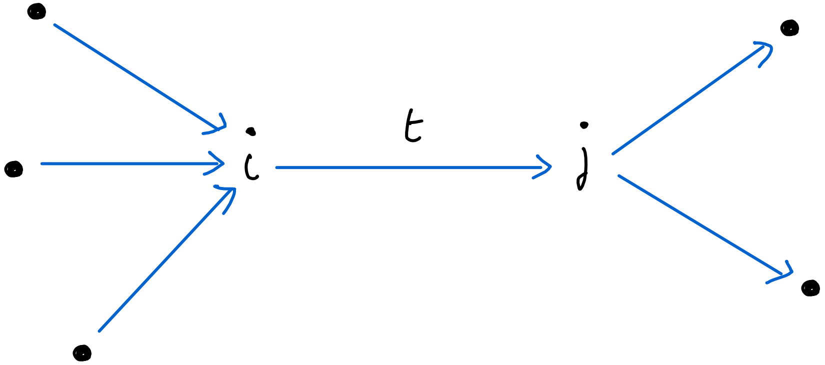


Repair monochromatic semi-connectedness?

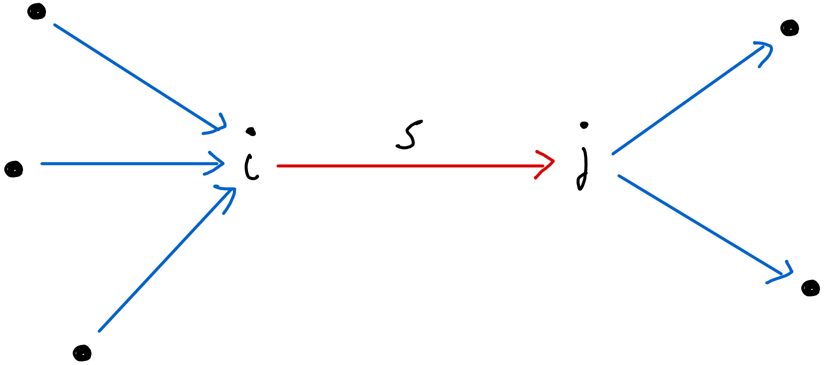
loop:
    $res.\mathtt{append}(c)$
    $i, j, t \leftarrow c.\mathtt{pop}()$
    if $t = 1$**: break**
    $t', \star \leftarrow \zeta.\mathtt{sep}(i, j, t)$
    $c.\mathtt{insert}(i, j, t', \star)$
    Repair monochromatic semi-connectedness



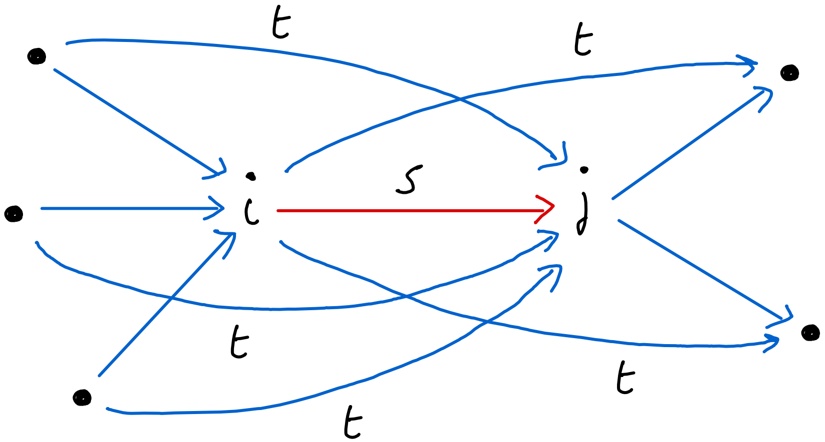
Repair monochromatic semi-connectedness!



Repair monochromatic semi-connectedness!



Repair monochromatic semi-connectedness!



Definition

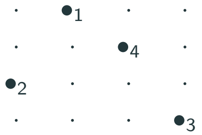
Let $\rho, \iota \in \mathfrak{S}_n$. We define

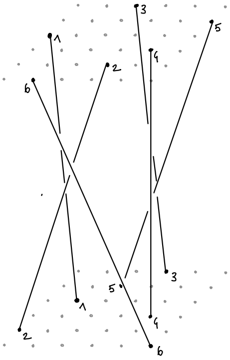
$$\omega_{\rho,\iota} = (\rho(1) + \mathbf{i}\iota(1), \ldots, \rho(n) + \mathbf{i}\iota(n)) \in OC_n$$

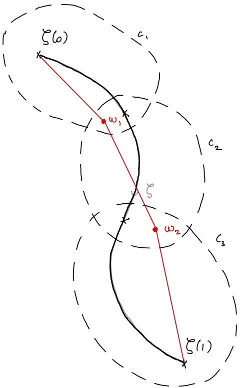
$$\rho = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 3 \end{pmatrix}$$

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

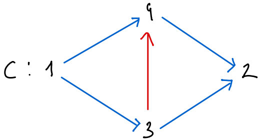
$\omega_{\rho,\iota} \quad =$







Given a cell, how do we find a $\omega_{\rho,\iota}$ in it?



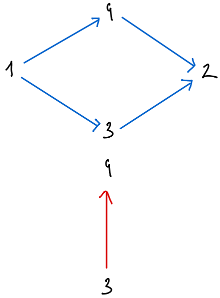
Problem

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

Solution

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

Given a cell, how do we find a $\omega_{\rho,\iota}$ in it?



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").

Given a cell, how do we find a $\omega_{\rho,\iota}$ in it?

$$\begin{tikzcd}
& 4 \arrow[dr] & \\
1 \arrow[ur] \arrow[dr] & & 2 \\
& 3 \arrow[ur] &
\end{tikzcd}$$

$\rightsquigarrow 1, 3, 4, 2$

$$\begin{tikzcd}
4 \\
3 \arrow[u]
\end{tikzcd}$$

$\rightsquigarrow 1, 2, 3, 4$

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").

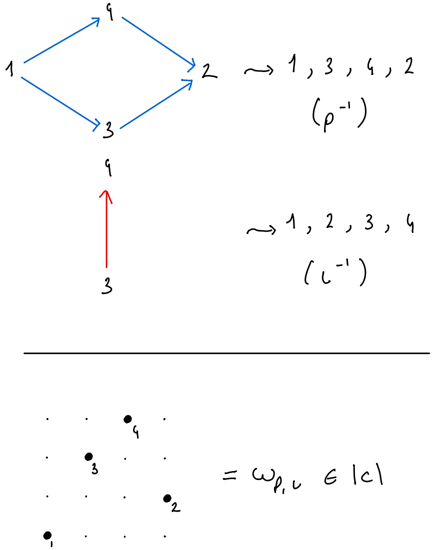
Given a cell, how do we find a $\omega_{\rho,\iota}$ in it?

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").



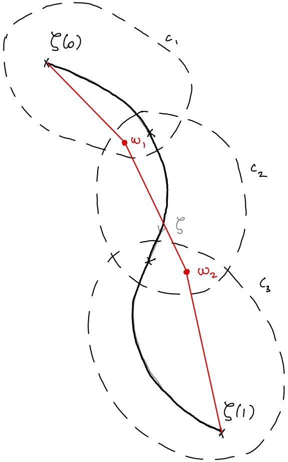
Algorithm

def `linearize`(ζ)

cells $\leftarrow \zeta$.path_to_cells()
res $\leftarrow [(1,1)]$ # $\omega_{1,1} \in (1 \xrightarrow{0} 2 \xrightarrow{0} \ldots \xrightarrow{0} n, \varnothing)$ the first element of *cells*
for each pair of successive cells c_i, c_{i+1} **in** *cells*:
    Compute ρ, ι such that $\omega_{\rho,\iota} \in |c_i| \cap |c_{i+1}|$
    res.append((ρ, ι))
$\sigma \leftarrow \zeta$.monodromy() # $\zeta_i(1) = \zeta_{\sigma(i)}(0)$
res.append((σ, ι))
return *res*

Proposition

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



Main property of `linearize`

$\dots$

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

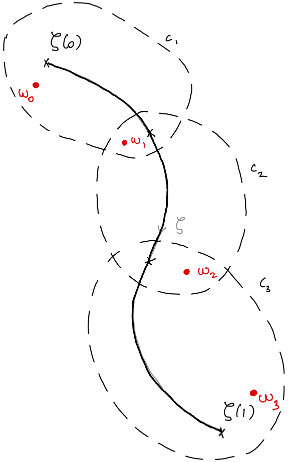
$\dots$

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

$\mathit{res}.\mathrm{append}((\sigma, \iota))$

Proposition

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



Main property of `linearize`

$\ldots$

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

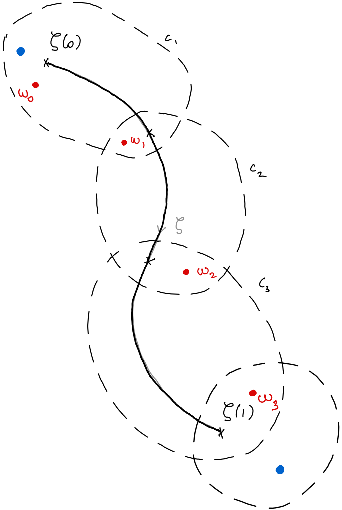
$\ldots$

$\sigma \leftarrow \zeta$.`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 ζ.`linearize()` are equal.



Main property of `linearize`

$\ldots$

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

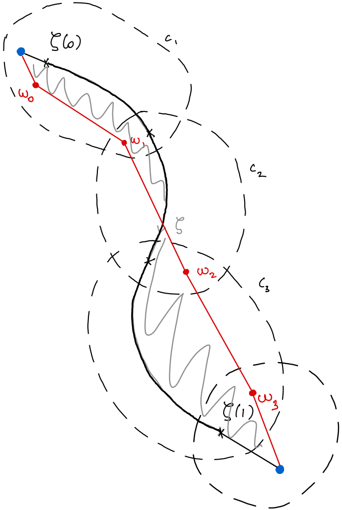
$\ldots$

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

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

Proposition

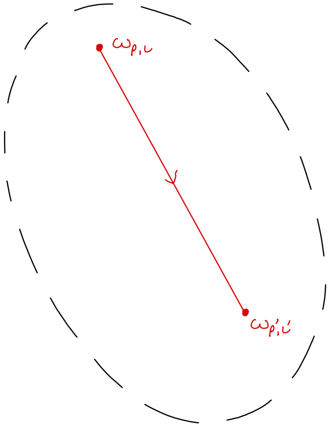
The braid associated to ζ and to the linear interpolation of the result of ζ.`linearize()` are equal.



Step 3: decomposition of the linearization in standard generators

Reduction

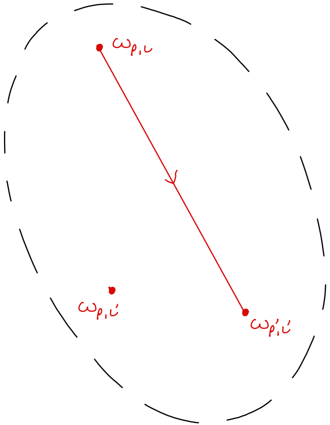
- Computing the braid associated to the whole linearization or to each piece and concatenating the results is equivalent



Step 3: decomposition of the linearization in standard generators

Reduction

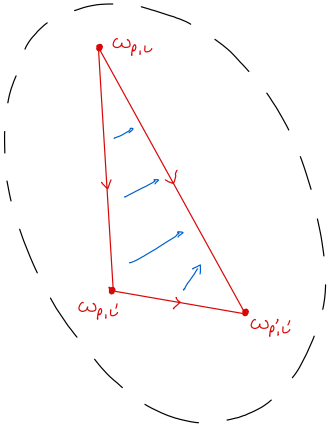
- Computing the braid associated to the whole linearization or to each piece and concatenating the results is equivalent
- Assume $\omega_{\rho,\iota}$ and $\omega_{\rho',\iota'}$ both lie in a m.s.c cell $c = (R, I)$. It means that ρ, ρ' extend R and ι, ι' extend I. **So $\omega_{\rho,\iota'}$ also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



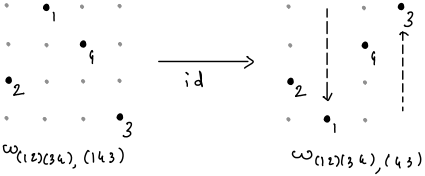
Step 3: decomposition of the linearization in standard generators

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 $\omega_{\rho,\iota} \to \omega_{\rho,\iota'}$ then the braid of $\omega_{\rho,\iota'} \to \omega_{\rho',\iota'}$



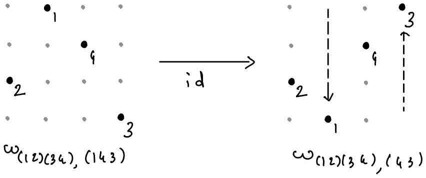
Step 3: decomposition of the linearization in standard generators

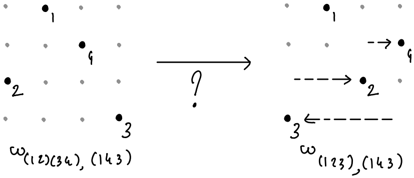


$\omega_{\rho,\iota} \to \omega_{\rho,\iota'}$

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

Step 3: decomposition of the linearization in standard generators

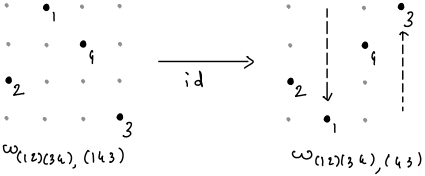


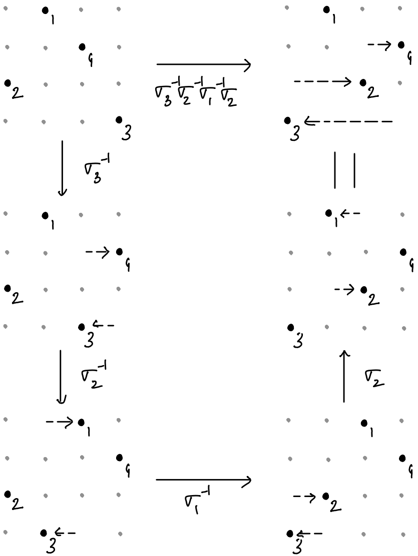


$\omega_{\rho,\iota} \to \omega_{\rho,\iota'}$

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

Step 3: decomposition of the linearization in standard generators





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

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

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

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 $1 \to \dots \to n$ has only $n-1$ edges.
- In the algorithm presented, we never decrease the number of edges.
- Optimization: before inserting and 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`
```

$$\sigma_{57}\sigma_{11}\sigma_{25}\sigma_{29}\sigma_{47}\sigma_{51}\sigma_{55}\sigma_{61}\sigma_{63}\sigma_{83}\sigma_{3}\sigma_{5}\sigma_{41}\sigma_{31}\sigma_{35}\sigma_{0}^{-1}\sigma_{1}\sigma_{0}\sigma_{37}\sigma_{73}\sigma_{97}\sigma_{98}\sigma_{27}\sigma_{49}\sigma_{87}\sigma_{97}^{-1}\sigma_{59}\sigma_{15}\sigma_{2}\sigma_{3}^{-1}\sigma_{96}^{-1}\sigma_{77}\sigma_{92}^{-1}$$
$$\sigma_{93}^{-1}\sigma_{81}\sigma_{92}\sigma_{24}\sigma_{72}^{-1}\sigma_{84}\sigma_{18}^{-1}\sigma_{26}\sigma_{94}^{-1}\sigma_{95}^{-1}\sigma_{10}\sigma_{88}\sigma_{87}^{-1}\sigma_{17}^{-1}\sigma_{24}^{-1}\sigma_{16}^{-1}\sigma_{25}\sigma_{24}\sigma_{82}\sigma_{15}^{-1}\sigma_{63}^{-1}\sigma_{86}^{-1}\sigma_{13}\sigma_{14}^{-1}$$
$$\sigma_{13}^{-1}\sigma_{6}^{-1}\sigma_{81}^{-1}\sigma_{65}\sigma_{87}^{-1}\sigma_{12}^{-1}\sigma_{73}^{-1}\sigma_{11}^{-1}\sigma_{88}^{-1}\sigma_{96}^{-1}\sigma_{93}\sigma_{94}\sigma_{4}\sigma_{5}^{-1}\sigma_{27}^{-1}\sigma_{10}^{-1}\sigma_{66}\sigma_{62}\sigma_{71}\sigma_{74}^{-1}\sigma_{6}^{-1}\sigma_{7}\sigma_{6}\sigma_{4}^{-1}\sigma_{15}\sigma_{9}^{-1}$$
$$\sigma_{93}\sigma_{23}\sigma_{16}^{-1}\sigma_{92}^{-1}\sigma_{8}^{-1}\sigma_{20}^{-1}\sigma_{7}^{-1}\sigma_{75}^{-1}\sigma_{6}^{-1}\sigma_{5}^{-1}\sigma_{4}^{-1}\sigma_{91}\sigma_{22}\sigma_{21}\sigma_{89}^{-1}\sigma_{20}\sigma_{73}\sigma_{85}\sigma_{90}^{-1}\sigma_{22}\sigma_{91}^{-1}\sigma_{3}^{-1}\sigma_{92}^{-1}\sigma_{6}\sigma_{93}^{-1}$$
$$\sigma_{82}^{-1}\sigma_{83}^{-1}\sigma_{2}^{-1}\sigma_{82}\sigma_{94}^{-1}\sigma_{95}^{-1}\sigma_{12}^{-1}\sigma_{67}\sigma_{0}^{-1}\sigma_{1}^{-1}\sigma_{0}^{-1}\sigma_{96}^{-1}\sigma_{35}^{-1}\sigma_{68}\sigma_{76}^{-1}\sigma_{77}^{-1}\sigma_{97}^{-1}\sigma_{14}\sigma_{15}^{-1}\sigma_{14}^{-1}\sigma_{98}^{-1}\sigma_{60}\sigma_{94}\sigma_{92}^{-1}$$
$$\sigma_{91}\sigma_{78}^{-1}\sigma_{13}^{-1}\sigma_{14}\sigma_{70}^{-1}\sigma_{69}\sigma_{70}\sigma_{59}^{-1}\sigma_{21}\sigma_{83}\sigma_{79}^{-1}\sigma_{80}^{-1}\sigma_{92}\sigma_{71}\sigma_{15}\sigma_{23}^{-1}\sigma_{17}^{-1}\sigma_{9}^{-1}\sigma_{18}\sigma_{10}\sigma_{19}\sigma_{81}^{-1}\sigma_{18}\sigma_{93}\sigma_{92}^{-1}$$
$$\sigma_{17}\sigma_{84}\sigma_{83}^{-1}\sigma_{82}^{-1}\sigma_{83}^{-1}\sigma_{84}^{-1}\sigma_{16}\sigma_{72}\sigma_{79}^{-1}\sigma_{12}^{-1}\sigma_{76}^{-1}\sigma_{13}\sigma_{85}^{-1}\sigma_{73}\sigma_{86}^{-1}\sigma_{36}\sigma_{74}\sigma_{81}\sigma_{87}^{-1}\sigma_{88}^{-1}\sigma_{15}\sigma_{89}^{-1}\sigma_{14}\sigma_{87}\sigma_{13}\sigma_{12}$$
$$\sigma_{17}^{-1}\sigma_{18}\sigma_{28}\sigma_{90}^{-1}\sigma_{91}^{-1}\sigma_{78}\sigma_{92}^{-1}\sigma_{93}^{-1}\sigma_{94}^{-1}\sigma_{95}^{-1}\sigma_{98}\sigma_{97}\sigma_{92}^{-1}\sigma_{96}^{-1}\sigma_{97}^{-1}\sigma_{24}\sigma_{98}^{-1}\sigma_{94}\sigma_{75}\sigma_{29}^{-1}\sigma_{15}\sigma_{76}\sigma_{88}\sigma_{87}^{-1}$$
$$\sigma_{33}\sigma_{95}\sigma_{94}\sigma_{93}\sigma_{92}\sigma_{11}\sigma_{77}\sigma_{78}\sigma_{68}\sigma_{67}^{-1}\sigma_{91}\sigma_{90}\sigma_{79}\sigma_{80}\sigma_{10}\sigma_{9}\sigma_{19}^{-1}\sigma_{79}^{-1}\sigma_{84}\sigma_{7}^{-1}\sigma_{8}\sigma_{7}\sigma_{30}\sigma_{78}\sigma_{6}\sigma_{8}\sigma_{5}\sigma_{91}\sigma_{31}^{-1}\sigma_{89}\sigma_{4}\sigma_{81}\sigma_{82}$$
$$\sigma_{47}^{-1}\sigma_{3}\sigma_{88}\sigma_{20}\sigma_{21}^{-1}\sigma_{83}\sigma_{81}\sigma_{84}\sigma_{85}\sigma_{76}^{-1}\sigma_{32}\sigma_{33}^{-1}\sigma_{86}\sigma_{2}\sigma_{1}\sigma_{94}\sigma_{87}\sigma_{77}^{-1}\sigma_{75}\sigma_{76}\sigma_{88}\sigma_{89}\sigma_{75}^{-1}\sigma_{90}\sigma_{91}\sigma_{66}\sigma_{65}^{-1}\sigma_{22}^{-1}\sigma_{85}^{-1}$$
$$\sigma_{84}\sigma_{92}\sigma_{93}\sigma_{94}\sigma_{95}\sigma_{23}\sigma_{96}\sigma_{97}\sigma_{24}\sigma_{82}^{-1}\sigma_{83}\sigma_{82}\sigma_{19}\sigma_{20}^{-1}\sigma_{34}^{-1}\sigma_{35}\sigma_{36}\sigma_{81}^{-1}\sigma_{74}^{-1}\sigma_{73}\sigma_{72}\sigma_{25}\sigma_{63}\sigma_{64}\sigma_{63}^{-1}\sigma_{73}^{-1}\sigma_{74}\sigma_{26}^{-1}$$
$$\sigma_{37}^{-1}\sigma_{38}^{-1}\sigma_{39}\sigma_{62}\sigma_{61}\sigma_{60}^{-1}\sigma_{59}^{-1}\sigma_{60}\sigma_{61}^{-1}\sigma_{23}^{-1}\sigma_{12}^{-1}\sigma_{13}\sigma_{58}\sigma_{57}^{-1}\sigma_{40}\sigma_{41}^{-1}\sigma_{22}\sigma_{14}\sigma_{15}^{-1}\sigma_{42}^{-1}\sigma_{43}\sigma_{56}\sigma_{55}^{-1}\sigma_{54}^{-1}$$
$$\sigma_{53}\sigma_{44}^{-1}\sigma_{72}^{-1}\sigma_{73}\sigma_{45}\sigma_{26}\sigma_{25}^{-1}\sigma_{84}^{-1}\sigma_{74}^{-1}\sigma_{75}\sigma_{86}\sigma_{76}^{-1}\sigma_{77}\sigma_{52}\sigma_{51}^{-1}\sigma_{38}\sigma_{85}\sigma_{50}\sigma_{49}^{-1}\sigma_{24}^{-1}\sigma_{23}\sigma_{46}^{-1}\sigma_{47}\sigma_{78}^{-1}\sigma_{79}\sigma_{98}$$
$$\sigma_{0}\sigma_{84}\sigma_{48}\sigma_{78}\sigma_{82}^{-1}\sigma_{80}\sigma_{30}^{-1}\sigma_{34}\sigma_{52}^{-1}\sigma_{74}\sigma_{76}\sigma_{90}^{-1}\sigma_{23}^{-1}\sigma_{24}\sigma_{36}^{-1}\sigma_{79}\sigma_{78}^{-1}\sigma_{4}^{-1}\sigma_{21}\sigma_{46}\sigma_{47}\sigma_{70}^{-1}\sigma_{46}^{-1}\sigma_{28}^{-1}\sigma_{68}^{-1}\sigma_{86}$$
$$\sigma_{92}\sigma_{6}\sigma_{23}\sigma_{54}\sigma_{66}^{-1}\sigma_{88}^{-1}\sigma_{94}^{-1}\sigma_{2}\sigma_{32}^{-1}\sigma_{49}\sigma_{50}^{-1}\sigma_{38}^{-1}\sigma_{10}\sigma_{26}^{-1}\sigma_{56}^{-1}\sigma_{58}^{-1}\sigma_{72}\sigma_{40}^{-1}\sigma_{44}\sigma_{51}^{-1}\sigma_{85}\sigma_{60}^{-1}\sigma_{52}\sigma_{12}\sigma_{96}^{-1}$$
$$\sigma_{45}\sigma_{44}^{-1}\sigma_{24}\sigma_{23}^{-1}\sigma_{17}\sigma_{53}\sigma_{54}^{-1}\sigma_{23}\sigma_{24}^{-1}\sigma_{77}\sigma_{76}^{-1}\sigma_{86}\sigma_{75}\sigma_{84}^{-1}\sigma_{42}\sigma_{43}\sigma_{42}^{-1}\sigma_{55}^{-1}\sigma_{56}\sigma_{62}^{-1}\sigma_{74}^{-1}\sigma_{25}^{-1}\sigma_{26}\sigma_{73}\sigma_{72}^{-1}\sigma_{57}^{-1}$$
$$\sigma_{58}\sigma_{38}\sigma_{60}\sigma_{61}\sigma_{60}^{-1}\sigma_{59}^{-1}\sigma_{22}\sigma_{49}^{-1}\sigma_{41}^{-1}\sigma_{14}^{-1}\sigma_{15}^{-1}\sigma_{14}\sigma_{23}^{-1}\sigma_{60}\sigma_{61}^{-1}\sigma_{40}\sigma_{13}\sigma_{12}^{-1}\sigma_{85}^{-1}\sigma_{62}\sigma_{39}\sigma_{38}^{-1}\sigma_{74}\sigma_{73}^{-1}$$
$$\sigma_{37}^{-1}\sigma_{60}^{-1}\sigma_{63}\sigma_{64}^{-1}\sigma_{36}\sigma_{35}\sigma_{34}^{-1}\sigma_{81}^{-1}\sigma_{65}^{-1}\sigma_{66}\sigma_{72}\sigma_{26}^{-1}\sigma_{25}\sigma_{82}\sigma_{83}\sigma_{24}\sigma_{25}^{-1}\sigma_{26}\sigma_{82}^{-1}\sigma_{84}^{-1}\sigma_{73}\sigma_{74}^{-1}\sigma_{32}\sigma_{33}\sigma_{32}^{-1}\sigma_{33}^{-1}$$
$$\sigma_{23}\sigma_{22}^{-1}\sigma_{19}^{-1}\sigma_{20}^{-1}\sigma_{19}\sigma_{97}\sigma_{96}\sigma_{31}^{-1}\sigma_{95}\sigma_{94}\sigma_{93}\sigma_{92}\sigma_{85}\sigma_{30}\sigma_{67}^{-1}\sigma_{68}\sigma_{8}^{-1}\sigma_{21}^{-1}\sigma_{20}\sigma_{91}\sigma_{90}\sigma_{75}^{-1}\sigma_{76}\sigma_{75}\sigma_{87}\sigma_{77}^{-1}\sigma_{88}$$
$$\sigma_{76}^{-1}\sigma_{89}\sigma_{88}\sigma_{87}\sigma_{86}\sigma_{85}\sigma_{81}\sigma_{47}^{-1}\sigma_{98}^{-1}\sigma_{97}^{-1}\sigma_{29}^{-1}\sigma_{28}\sigma_{96}^{-1}\sigma_{84}\sigma_{94}\sigma_{95}^{-1}\sigma_{94}^{-1}\sigma_{93}^{-1}\sigma_{83}\sigma_{82}\sigma_{81}\sigma_{91}\sigma_{92}^{-1}\sigma_{91}^{-1}\sigma_{90}^{-1}$$
$$\sigma_{78}\sigma_{88}\sigma_{89}^{-1}\sigma_{88}^{-1}\sigma_{79}^{-1}\sigma_{87}^{-1}\sigma_{82}^{-1}\sigma_{86}^{-1}\sigma_{1}\sigma_{2}\sigma_{59}^{-1}\sigma_{60}\sigma_{77}\sigma_{18}\sigma_{17}^{-1}\sigma_{85}^{-1}\sigma_{84}^{-1}\sigma_{83}^{-1}\sigma_{82}^{-1}\sigma_{78}^{-1}\sigma_{3}\sigma_{4}\sigma_{88}\sigma_{80}\sigma_{89}^{-1}$$
$$\sigma_{18}^{-1}\sigma_{19}^{-1}\sigma_{18}\sigma_{19}\sigma_{69}^{-1}\sigma_{79}\sigma_{84}\sigma_{16}^{-1}\sigma_{78}\sigma_{96}\sigma_{77}\sigma_{70}\sigma_{81}^{-1}\sigma_{5}\sigma_{6}\sigma_{80}^{-1}\sigma_{7}\sigma_{8}\sigma_{0}^{-1}\sigma_{1}^{-1}\sigma_{94}^{-1}\sigma_{79}^{-1}\sigma_{75}^{-1}\sigma_{76}\sigma_{75}\sigma_{78}^{-1}\sigma_{17}^{-1}$$
$$\sigma_{15}\sigma_{2}^{-1}\sigma_{3}^{-1}\sigma_{11}\sigma_{10}^{-1}\sigma_{9}\sigma_{10}\sigma_{77}^{-1}\sigma_{90}^{-1}\sigma_{76}^{-1}\sigma_{11}\sigma_{16}\sigma_{4}^{-1}\sigma_{5}^{-1}\sigma_{22}^{-1}\sigma_{6}^{-1}\sigma_{7}^{-1}\sigma_{74}\sigma_{73}\sigma_{14}\sigma_{13}^{-1}\sigma_{14}^{-1}\sigma_{15}^{-1}\sigma_{14}\sigma_{8}^{-1}\sigma_{75}^{-1}$$
$$\sigma_{74}^{-1}\sigma_{9}^{-1}\sigma_{10}^{-1}\sigma_{23}^{-1}\sigma_{72}\sigma_{21}\sigma_{26}^{-1}\sigma_{12}\sigma_{13}\sigma_{71}\sigma_{22}\sigma_{73}^{-1}\sigma_{67}\sigma_{70}\sigma_{69}\sigma_{70}\sigma_{25}\sigma_{89}^{-1}\sigma_{72}^{-1}\sigma_{14}\sigma_{15}\sigma_{93}^{-1}\sigma_{92}^{-1}\sigma_{94}\sigma_{93}\sigma_{92}$$
$$\sigma_{71}^{-1}\sigma_{16}\sigma_{70}^{-1}\sigma_{17}\sigma_{88}^{-1}\sigma_{95}\sigma_{96}^{-1}\sigma_{11}^{-1}\sigma_{12}^{-1}\sigma_{15}^{-1}\sigma_{27}^{-1}\sigma_{13}^{-1}\sigma_{14}^{-1}\sigma_{15}^{-1}\sigma_{16}^{-1}\sigma_{94}\sigma_{95}^{-1}\sigma_{18}\sigma_{19}\sigma_{86}\sigma_{15}^{-1}\sigma_{26}\sigma_{68}\sigma_{67}$$
$$\sigma_{11}^{-1}\sigma_{85}\sigma_{69}^{-1}\sigma_{13}\sigma_{14}\sigma_{68}^{-1}\sigma_{87}^{-1}\sigma_{33}\sigma_{17}^{-1}\sigma_{86}^{-1}\sigma_{18}^{-1}\sigma_{82}^{-1}\sigma_{83}\sigma_{4}^{-1}\sigma_{5}^{-1}\sigma_{6}^{-1}\sigma_{94}^{-1}\sigma_{81}^{-1}\sigma_{82}\sigma_{4}\sigma_{5}\sigma_{6}\sigma_{7}\sigma_{91}\sigma_{93}^{-1}\sigma_{10}\sigma_{92}^{-1}$$
$$\sigma_{88}\sigma_{21}^{-1}\sigma_{89}^{-1}\sigma_{23}\sigma_{20}\sigma_{21}\sigma_{22}\sigma_{21}\sigma_{20}\sigma_{87}\sigma_{88}^{-1}\sigma_{23}\sigma_{19}^{-1}\sigma_{6}^{-1}\sigma_{20}^{-1}\sigma_{21}^{-1}\sigma_{22}^{-1}\sigma_{5}^{-1}\sigma_{4}\sigma_{93}^{-1}\sigma_{20}\sigma_{19}^{-1}\sigma_{20}^{-1}\sigma_{94}\sigma_{65}$$
$$\sigma_{13}\sigma_{95}\sigma_{96}^{-1}\sigma_{2}\sigma_{3}\sigma_{2}^{-1}\sigma_{1}\sigma_{0}^{-1}\sigma_{1}^{-1}\sigma_{3}^{-1}\sigma_{7}^{-1}\sigma_{9}^{-1}\sigma_{13}^{-1}\sigma_{17}^{-1}\sigma_{19}^{-1}\sigma_{21}^{-1}\sigma_{23}^{-1}\sigma_{33}^{-1}\sigma_{35}^{-1}\sigma_{39}^{-1}\sigma_{43}^{-1}\sigma_{45}^{-1}\sigma_{53}^{-1}\sigma_{63}^{-1}$$
$$\sigma_{65}^{-1}\sigma_{67}^{-1}\sigma_{69}^{-1}\sigma_{71}^{-1}\sigma_{75}^{-1}\sigma_{79}^{-1}\sigma_{85}^{-1}\sigma_{87}^{-1}\sigma_{89}^{-1}\sigma_{91}^{-1}\sigma_{95}^{-1}\sigma_{97}^{-1}\sigma_{98}$$