A symbolic-numeric algorithm for genus computation

Mădălina Hodorog
Supervisor: Prof. Dr. Josef Schicho

Johann Radon Institute for Computational and Applied Mathematics,
Austrian Academy of Sciences,
Research Institute for Symbolic Computation,
Johannes Kepler University Linz, Austria

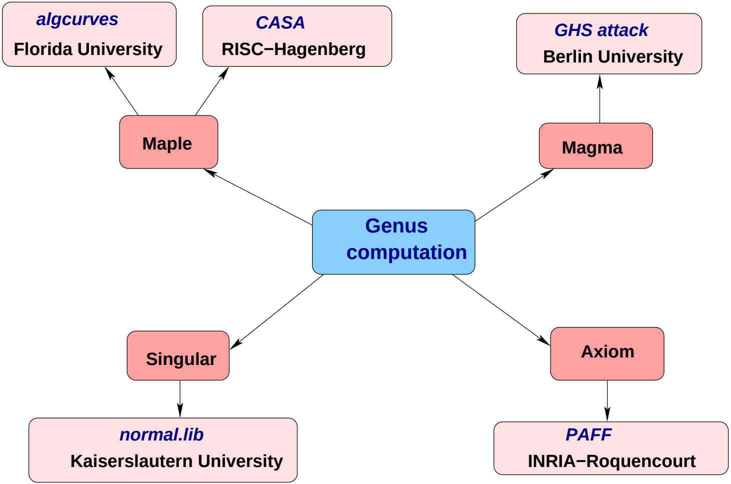
May 4, 2009

Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

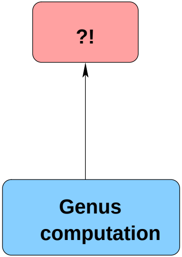
Table of contents

Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

Exact Algorithms:

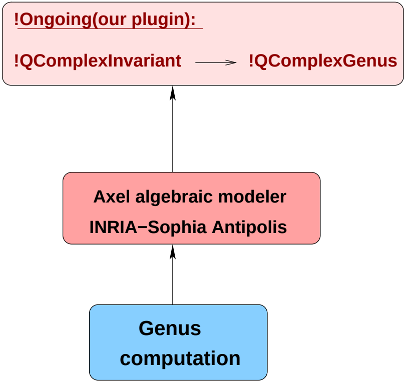


Numeric Algorithms:



Numeric Algorithms:

DK Project: Symbolic-Numeric techniques for genus computation and parametrization (project leader: Prof. Dr. Josef Schicho).



1. Motivation
2. Describing the Problem
   - What?
3. Solving the problem
   - How?
4. Current results
5. Conclusion and future work

Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

What?

- Input:
  - $\mathbb{C}$ field of complex numbers;
  - $F \in \mathbb{C}[z, w]$ irreducible with coefficients of limited accuracy [1];
  - $C = \{(z, w) \in \mathbb{C}^2 | F(z, w) = 0\} =$
    $= \{(x, y, u, v) \in \mathbb{R}^4 | F(x + iy, u + iv) = 0\}$ complex algebraic curve
    (d is the degree, $Sing(C)$ is the set of singularities);
- Output:
  - approximate $genus(C)$ s.t.

$$genus(C) = \frac{1}{2}(d-1)(d-2) - \sum_{P \in Sing(C)} \delta\text{-invariant}(P);$$

[1] For now: symbolic coefficients

1. Motivation

2. Describing the Problem
   - What?

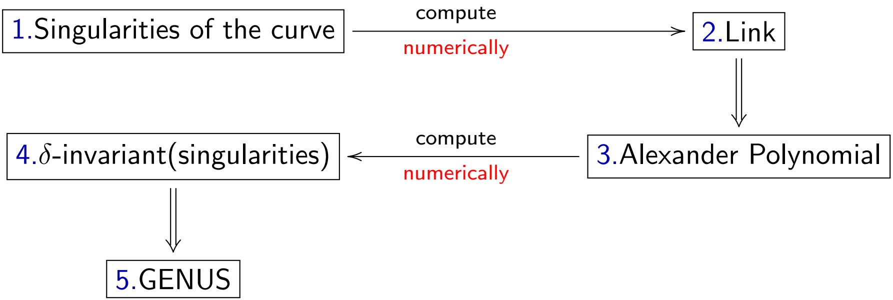
3. Solving the problem
   - How?

4. Current results

5. Conclusion and future work

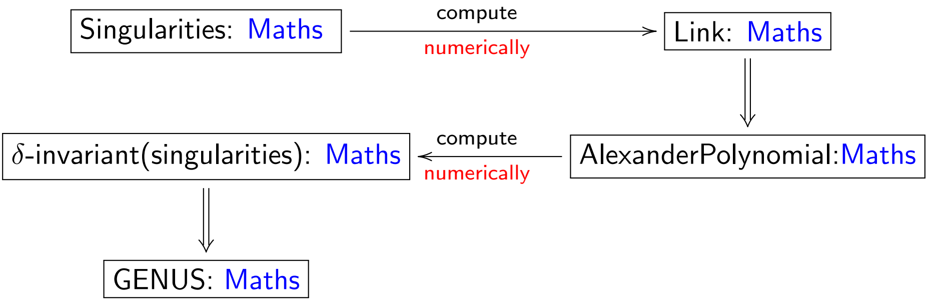
How?

- Strategy for computing the genus



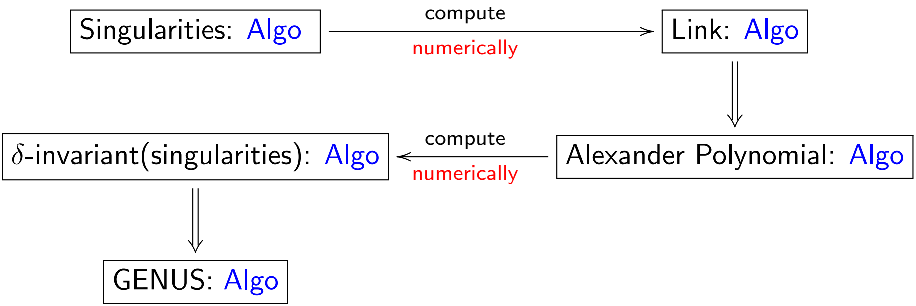
How?

- Method for computing the genus



How?

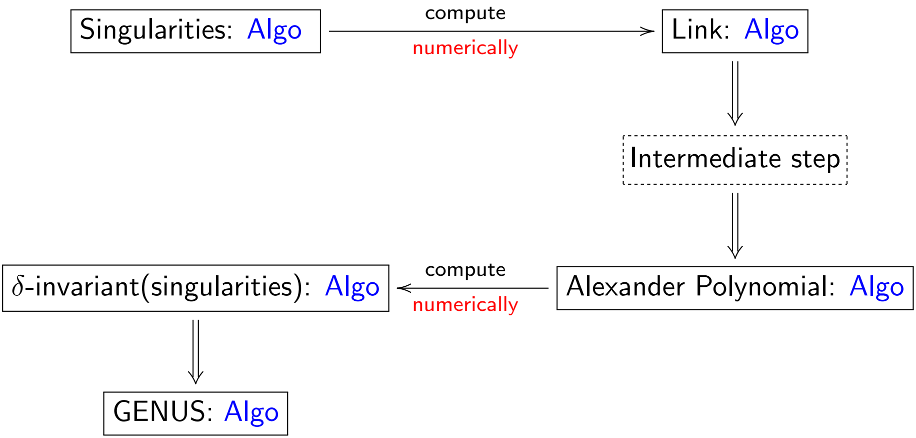
- Algorithm for the method



Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

How?

- Algorithm for the method



Solving the problem

Implementation of the algorithm

- (*Mathematica* computer algebra system)
- *Axel* algebraic geometric modeler [a]
  - developed by *Galaad* team (INRIA Sophia-Antipolis);
  - written in Qt scripting language;
  - topology of implicit curves;
  - intersections of implicit surfaces.

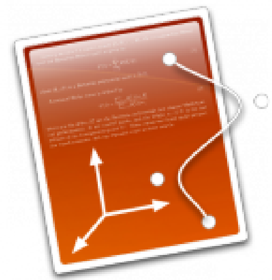
[a] Acknowledgements: B. Mourrain, J. Wintz

Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

Solving the problem

Implementation of the algorithm

- (*Mathematica* computer algebra system)
- *Axel* algebraic geometric modeler [a]
  - developed by *Galaad* team (INRIA Sophia-Antipolis);
  - written in Qt scripting language;
  - topology of implicit curves;
  - intersections of implicit surfaces.



[a] Acknowledgements: B. Mourrain, J. Wintz

Solving the problem

Implementation of the algorithm

- (*Mathematica* computer algebra system)
- *Axel* algebraic geometric modeler [a]
    - developed by *Galaad* team (INRIA Sophia-Antipolis);
    - written in Qt scripting language;
    - topology of implicit curves;
    - intersections of implicit surfaces.

[a] Acknowledgements: B. Mourrain, J. Wintz

Solving the problem

Implementation of the algorithm

- (*Mathematica* computer algebra system)
- *Axel* algebraic geometric modeler [a]
  - developed by *Galaad* team (INRIA Sophia-Antipolis);
  - written in Qt scripting language;
  - topology of implicit curves;
  - intersections of implicit surfaces.

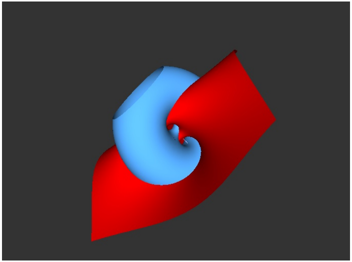
[a] Acknowledgements: B. Mourrain, J. Wintz

Doctoral Program Computational Mathematics Numerical Analysis and Symbolic Computation

Solving the problem

Implementation of the algorithm

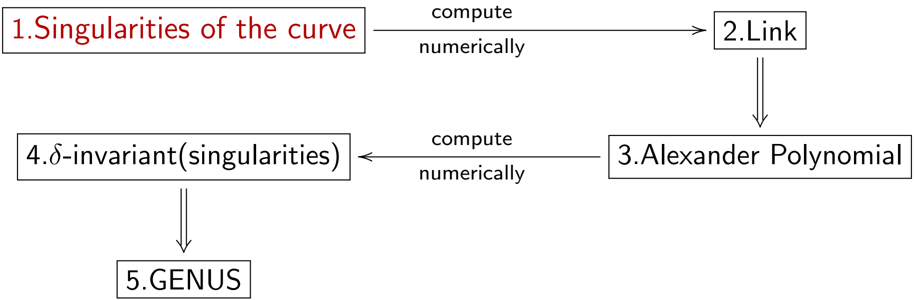
- (*Mathematica* computer algebra system)
- *Axel* algebraic geometric modeler [a]
    - developed by *Galaad* team (INRIA Sophia-Antipolis);
    - written in Qt scripting language;
    - topology of implicit curves;
    - intersections of implicit surfaces.



[a] Acknowledgements: B. Mourrain, J. Wintz

Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

First


1.Singularities of the curve
compute
numerically
2.Link
3.Alexander Polynomial
compute
numerically
4.δ-invariant(singularities)
5.GENUS

Computing the singularities of the curve

- Input:
  - $F \in \mathbb{C}[z, w]$
  - $C = \{(z, w) \in \mathbb{C}^2 | F(z, w) = 0\}$
- Output:
  - $S = \{(z_0, w_0) \in \mathbb{C}^2 | F(z_0, w_0) = 0, \frac{\delta F}{\delta z}(z_0, w_0) = 0, \frac{\delta F}{\delta w}(z_0, w_0) = 0\}$

Method: $\Rightarrow$ solve overdeterminate system of polynomial equations in $\mathbb{C}^2$:

$$\begin{cases} F(z_0, w_0) = 0 \\ \frac{\delta F}{\delta z}(z_0, w_0) = 0 \\ \frac{\delta F}{\delta w}(z_0, w_0) = 0 \end{cases}, \tag{1}$$

Computing the singularities of the curve

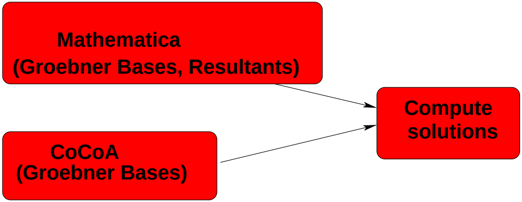
or in $\mathbb{R}^4$: $F(z,w) = F(x+iy, u+iv) = s(x,y,u,v) + it(x,y,u,v)$

$$\begin{cases} s(x_0,y_0,u_0,v_0) = 0 \\ t(x_0,y_0,u_0,v_0) = 0 \\ \\ \dfrac{\delta s}{\delta x}(x_0,y_0,u_0,v_0) = 0 \\ \\ \dfrac{\delta t}{\delta x}(x_0,y_0,u_0,v_0) = 0 \\ \\ \dfrac{\delta s}{\delta u}(x_0,y_0,u_0,v_0) = 0 \\ \\ \dfrac{\delta t}{\delta u}(x_0,y_0,u_0,v_0) = 0 \end{cases} , \tag{2}$$

Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

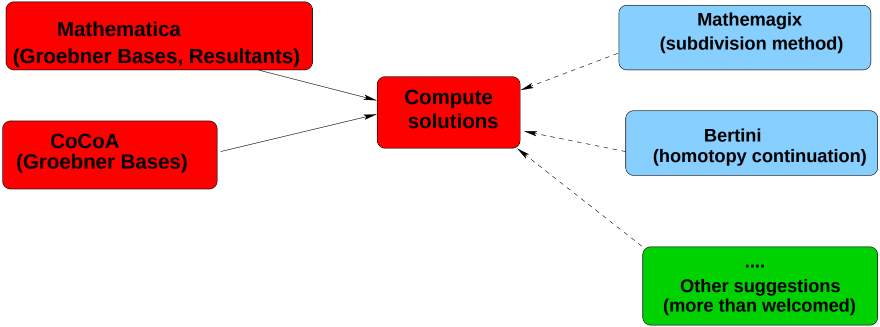
Computing the singularities of the curve

Using numeric input polynomials



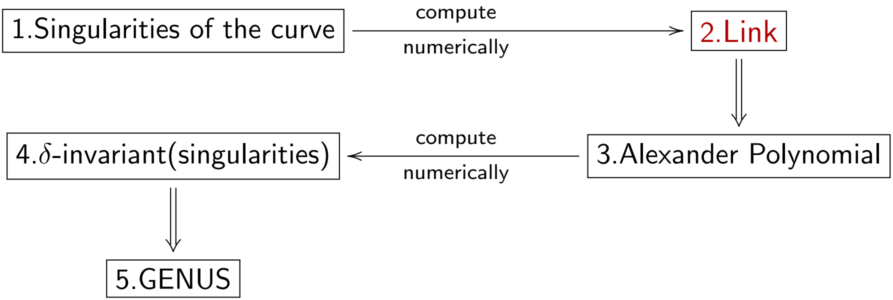
Computing the singularities of the curve

Using numeric input polynomials



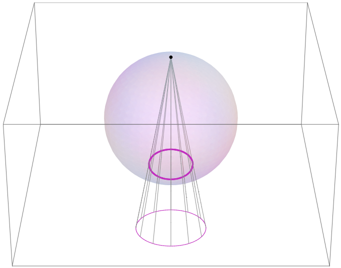
Note: so far an open problem.

Next


1.Singularities of the curve
compute
numerically
2.Link
3.Alexander Polynomial
compute
numerically
4.δ-invariant(singularities)
5.GENUS

Computing the link of the singularity

- Why the link of a singularity?
  - helps in understanding the topology of a complex curve near a singularity;
- How do we compute the link?
  - use stereographic projection;



-

Computing the link of the singularity

Method (based on Milnor's results)

1. Let $C = \{(x,y,u,v) \in \mathbb{R}^4 | F(x+iy, u+iv) = 0\} \subset \mathbb{C}^2 \cong \mathbb{R}^4$, with $\left(F(0,0), \frac{\delta F}{\delta z}(0,0), \frac{\delta F}{\delta w}(0,0)\right) = (0,0,0)$, where $z = x+iy, w = u+iv.$

2. Consider $S^3 = \{(x,y,u,v) \in \mathbb{R}^4 | x^2+y^2+u^2+w^2 = \epsilon^2\} \subset \mathbb{R}^4$ and $X = C \bigcap S^3 = \{(x,y,u,v) \in \mathbb{R}^4 | F(x,y,u,v) = 0, x^2+y^2+u^2+w^2 = \epsilon^2\}.$

3. For $P(0,0,0,\epsilon) \in S^3 \setminus C$, construct
$f : S^3 \setminus \{P\} \subset \mathbb{R}^4 \to \mathbb{R}^3, (x,y,u,v) \to (a,b,c) = (\frac{x}{\epsilon - v}, \frac{y}{\epsilon - v}, \frac{u}{\epsilon - v})$
$f(X) = \{(a,b,c) \in \mathbb{R}^3 | \exists (x,y,u,v) \in C \bigcap S^3 : (a,b,c) = f(x,y,u,v)\}$
$f(X)$ is a link.

Computing the link of the singularity

Method (next)

3. $f(X) = \{(a,b,c) \in \mathbb{R}^3 | \exists (x,y,u,v) \in C \bigcap S^3 : (a,b,c) = f(x,y,u,v)\}$

$$f(X) = \{(a,b,c) \in \mathbb{R}^3 | \exists (x,y,u,v) = f^{-1}(a,b,c) \in C \bigcap S^3\}$$

4. Compute $f^{-1} : \mathbb{R}^3 \to S^3 \setminus \{P\}$

$$(a,b,c) \to (x,y,u,v) = \left(\frac{2a\epsilon}{1+a^2+b^2+c^2}, \frac{2b\epsilon}{1+a^2+b^2+c^2}, \frac{2c\epsilon}{1+a^2+b^2+c^2}, \frac{\epsilon(a^2+b^2+c^2-1)}{1+a^2+b^2+c^2}\right)$$

5. Get $\begin{array}{l} f(X) = \Big\{(a,b,c) \in \mathbb{R}^3 | \\ F\left(\frac{2a\epsilon}{1+a^2+b^2+c^2}, \frac{2b\epsilon}{1+a^2+b^2+c^2}, \frac{2c\epsilon}{1+a^2+b^2+c^2}, \frac{\epsilon(a^2+b^2+c^2-1)}{1+a^2+b^2+c^2}\right) = 0\Big\} \end{array} \Leftrightarrow$

$$f(X) = \{(a,b,c) \in \mathbb{R}^3 | ReF(...) = 0, ImF(...) = 0\}$$

Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

Computing the link of the singularity

Method (next)

3. $f(X) = \{(a,b,c) \in \mathbb{R}^3 | \exists (x,y,u,v) \in C \bigcap S^3 : (a,b,c) = f(x,y,u,v)\}$
$$f(X) = \{(a,b,c) \in \mathbb{R}^3 | \exists (x,y,u,v) = f^{-1}(a,b,c) \in C \bigcap S^3\}$$

4. Compute $f^{-1} : \mathbb{R}^3 \to S^3 \setminus \{P\}$
$$(a,b,c) \to (x,y,u,v) = \left(\frac{2a\epsilon}{1+a^2+b^2+c^2}, \frac{2b\epsilon}{1+a^2+b^2+c^2}, \frac{2c\epsilon}{1+a^2+b^2+c^2}, \frac{\epsilon(a^2+b^2+c^2-1)}{1+a^2+b^2+c^2}\right)$$

5. Get
$$f(X) = \Big\{(a,b,c) \in \mathbb{R}^3 | \; F\left(\frac{2a\epsilon}{1+a^2+b^2+c^2}, \frac{2b\epsilon}{1+a^2+b^2+c^2}, \frac{2c\epsilon}{1+a^2+b^2+c^2}, \frac{\epsilon(a^2+b^2+c^2-1)}{1+a^2+b^2+c^2}\right) = 0\Big\}$$

Compute B s.t.
$$f(X) = \{(a,b,c) \in B \subset \mathbb{R}^3 | ReF(...) = 0, ImF(...) = 0\}$$ is a link

Doctoral Program Computational Mathematics Numerical Analysis and Symbolic Computation

Computing the link of the singularity

Method (next)

6. For $f(X) = \left\{(a,b,c) \in B \subset \mathbb{R}^3 | ReF(...) = 0, ImF(...) = 0\right\}$ link

find $B = max\left\{||f(p)||_\infty, p \in S^3 \bigcap C\right\} \leq max\{||f(p)||_2, p \in S^3 \bigcap C\}$

...

7. Compute

$$v_0 = max\{v : (x,y,u,v) \in S^3 \bigcap C\} \text{ s.t. } v \text{ is solution for}$$

$$\begin{cases} x^2 + y^2 + u^2 + v^2 - \epsilon^2 = 0 \\ ReF(x+iy, u+iv) = 0 \\ ImF(x+iy, u+iv) = 0 \end{cases},$$

Doctoral Program Computational Mathematics Numerical Analysis and Symbolic Computation

Computing the link of the singularity

Method (next)

6. For $f(X) = \{(a,b,c) \in B \subset \mathbb{R}^3 | ReF(...) = 0, ImF(...) = 0\}$ link

find $B = max\{||f(p)||_\infty, p \in S^3 \bigcap C\} \leq max\{||f(p)||_2, p \in S^3 \bigcap C\}$
...

7. Compute

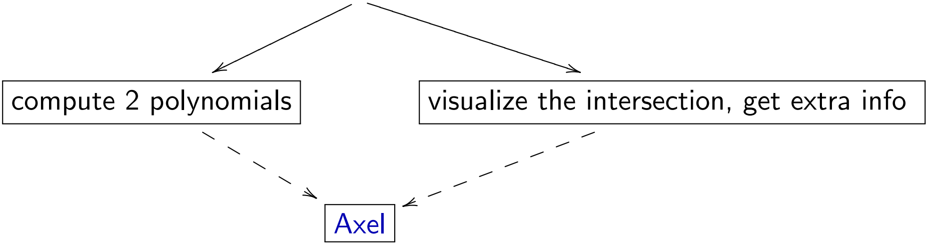
$$b = \sqrt{\frac{\epsilon + v_0}{\epsilon - v_0}}$$

Get $B = [-b, b]^3$

Computing the link of the singularity

Method (summary)

$$f(X) = \{(a,b,c) \in B \subset \mathbb{R}^3 | ReF(...) = 0, ImF(...) = 0\}$$



Computing the link of the singularity

Why Axel?

Axel computes the topology of implicit curves in $\mathbb{R}^3$.

In our case:

- Input:
  - $ReF(...), ImF(...) \in \mathbb{R}[a, b, c]$
  - $C = \{(a, b, c) \in \mathbb{R}^3 | ReF(...) = 0, ImF(...) = 0\}$
  - $B = [-b, b] \times [-b, b] \times [-b, b], \epsilon \geq 0$
- Output:
  - $Graph(C) = \langle \mathcal{V}, \mathcal{E} \rangle$ with
    $\mathcal{V} = \{p = (m, n, q) \in \mathbb{R}^3\}$
    $\mathcal{E} = \{(i, j) | i, j \in \mathcal{V}\}$
- s.t. $Graph(C) \cong_{isotopic} C$

Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

Computing the link of the singularity

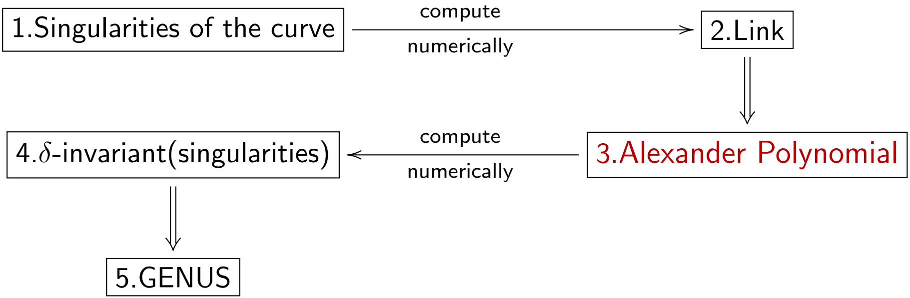
Test experiments (with Axel)

Equation	Tests on ϵ					
	ϵ=0.5		ϵ=1.0		ϵ=4.3	
	$[-b,b]^3$	link	$[-b,b]^3$	link	$[-b,b]^3$	link
$z^2 - w^2$	2.41421	**Hopf link**	2.41421	Hopf link	2.41421	Hopf link
$z^2 - w^3$	3.38298	**Trefoil knot**	2.67567	Trefoil knot	1.84639	Trefoil knot
$z^2 - w^2 - w^3$	2.37636	**Hopf link**	2.28464	Curve one singularity	2.24247	Trefoil knot

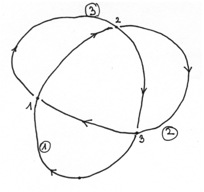
V.I. Arnold's results: $Top(z^2 - w^2 - w^3) \cong Top(z^2 - w^2)$

Note: solved problem.

Next


1.Singularities of the curve
compute
numerically
2.Link
3.Alexander Polynomial
compute
numerically
4.δ-invariant(singularities)
5.GENUS

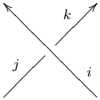
Computing the Alexander polynomial of the link



$$M(L) = \left(\begin{array}{c|cccc} & type & label_i & label_j & label_k \\ \hline c_1 & +1 & 1 & 2 & 3 \\ & & & & \\ & & & & \end{array} \right)$$

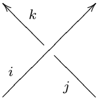
$$P(L) = \left(\quad \right)$$

Lefthanded (-1)



i	j	k
1-t	t	-1

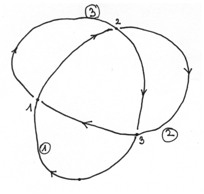
Righthanded (+1)



i	j	k
1-t	-1	t

Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

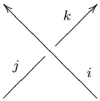
Computing the Alexander polynomial of the link



$$M(L) = \left(\begin{array}{c|cccc} & type & label_i & label_j & label_k \\ \hline c_1 & +1 & 1 & 2 & 3 \\ & & 1-t & -1 & t \\ & & & & \end{array}\right)$$

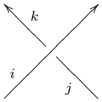
$$P(L) = \begin{pmatrix} 1 & 2 & 3 \\ 1-t & -1 & t \\ & & \end{pmatrix}$$

Lefthanded (-1)



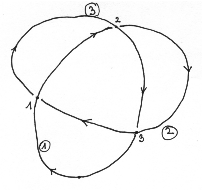
i	j	k
1-t	t	-1

Righthanded (+1)



i	j	k
1-t	-1	t

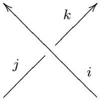
Computing the Alexander polynomial of the link



$$M(L) = \left(\begin{array}{c|cccc} & type & label_i & label_j & label_k \\ \hline c_2 & +1 & 3 & 1 & 2 \end{array} \right)$$

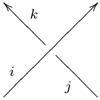
$$P(L) = \left(\quad \right)$$

Lefthanded (-1)



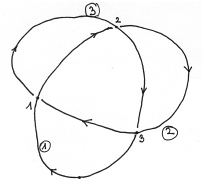
i	j	k
1-t	t	-1

Righthanded (+1)



i	j	k
1-t	-1	t

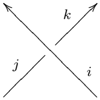
Computing the Alexander polynomial of the link



$$M(L) = \left(\begin{array}{c|cccc} & type & label_i & label_j & label_k \\ \hline & & & & \\ c_2 & +1 & 3 & 1 & 2 \\ & & 1-t & -1 & t \end{array} \right)$$

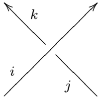
$$P(L) = \begin{pmatrix} 3 & 1 & 2 \\ 1-t & -1 & t \\ & & \end{pmatrix}$$

Lefthanded (-1)



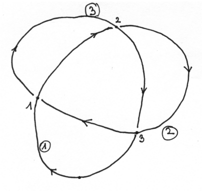
i	j	k
1-t	t	-1

Righthanded (+1)



i	j	k
1-t	-1	t

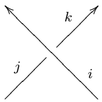
Computing the Alexander polynomial of the link



$$M(L) = \left(\begin{array}{c|cccc} & type & label_i & label_j & label_k \\ \hline c_2 & \color{blue}{+1} & 3 & 1 & 2 \\ & & \color{blue}{1-t} & \color{blue}{-1} & \color{blue}{t} \end{array}\right)$$

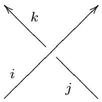
$$P(L) = \begin{pmatrix} \color{blue}{1} & \color{blue}{2} & \color{blue}{3} \\ -1 & t & 1-t \end{pmatrix}$$

Lefthanded (-1)



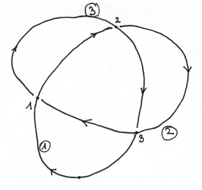
i	j	k
1-t	t	-1

Righthanded (+1)

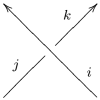


i	j	k
1-t	-1	t

Computing the Alexander polynomial of the link

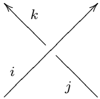


Lefthanded (-1)



i	j	k
1-t	t	-1

Righthanded (+1)

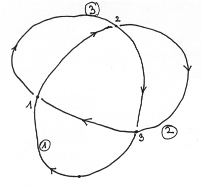


i	j	k
1-t	-1	t

$$M(L) = \left(\begin{array}{c|cccc} & type & label_i & label_j & label_k \\ \hline & & & & \\ c_3 & +1 & 2 & 3 & 1 \end{array} \right)$$

$$P(L) = \left(\quad \right)$$

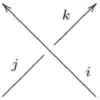
Computing the Alexander polynomial of the link



$$M(L) = \left(\begin{array}{c|cccc} & type & label_i & label_j & label_k \\ \hline & & & & \\ c_3 & +1 & 2 & 3 & 1 \\ & & 1-t & -1 & t \end{array}\right)$$

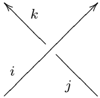
$$P(L) = \left(\begin{array}{ccc} & & \\ 2 & 3 & 1 \\ 1-t & -1 & t \end{array}\right)$$

Lefthanded (-1)



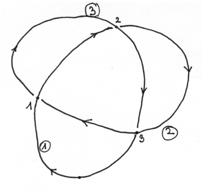
i	j	k
1-t	t	-1

Righthanded (+1)



i	j	k
1-t	-1	t

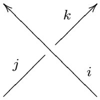
Computing the Alexander polynomial of the link



$$M(L) = \left(\begin{array}{c|cccc} & type & label_i & label_j & label_k \\ \hline & & & & \\ c_3 & +1 & 2 & 3 & 1 \\ & & 1-t & -1 & t \end{array} \right)$$

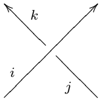
$$P(L) = \left(\begin{array}{ccc} & & \\ 1 & 2 & 3 \\ t & 1-t & -1 \end{array} \right)$$

Lefthanded (-1)



i	j	k
1-t	t	-1

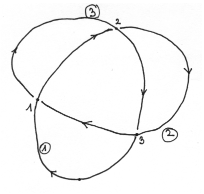
Righthanded (+1)



i	j	k
1-t	-1	t

Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

Computing the Alexander polynomial of the link

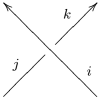


$$M(L) = \left(\begin{array}{c|cccc} & type & label_i & label_j & label_k \\ \hline c_1 & +1 & 1 & 2 & 3 \\ c_2 & +1 & 3 & 1 & 2 \\ c_3 & +1 & 2 & 3 & 1 \end{array}\right)$$

$$P(L) = \begin{pmatrix} 1-t & -1 & t \\ -1 & t & 1-t \\ t & 1-t & -1 \end{pmatrix}$$

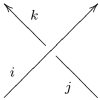
$$\Delta(L) := \Delta(t) = det(P(M)) = t^2 - t + 1$$

Lefthanded (-1)



i	j	k
1-t	t	-1

Righthanded (+1)



i	j	k
1-t	-1	t

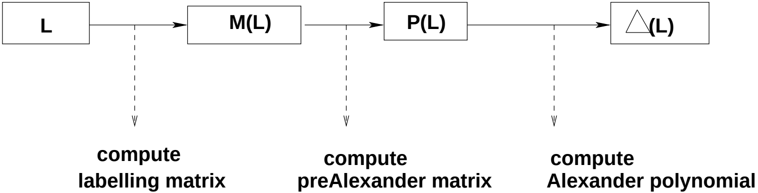
Computing the Alexander polynomial of the link

- Input:
  - $L = K_1 \cup ... \cup K_m$ with n - crossings
  - $D(L)$- oriented diagram of L
- Output:
  - $\Delta_L(t_1, ...t_m) \in \mathbb{Z}[t_1^{\pm 1}, ..., t_m^{\pm 1}]$
- Method: consists of several steps

- Need D(L)!

Computing the Alexander polynomial of the link

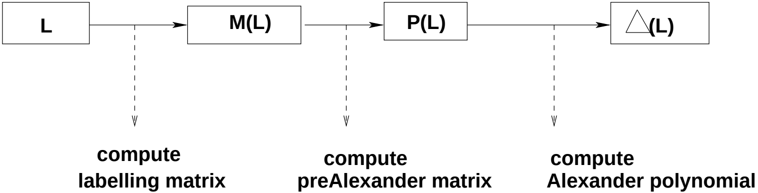
- Input:
  - $L = K_1 \cup ... \cup K_m$ with n - crossings
  - $D(L)$- oriented diagram of L
- Output:
  - $\Delta_L(t_1, ...t_m) \in \mathbb{Z}[t_1^{\pm 1}, ..., t_m^{\pm 1}]$
- Method: consists of several steps



- Need D(L)!

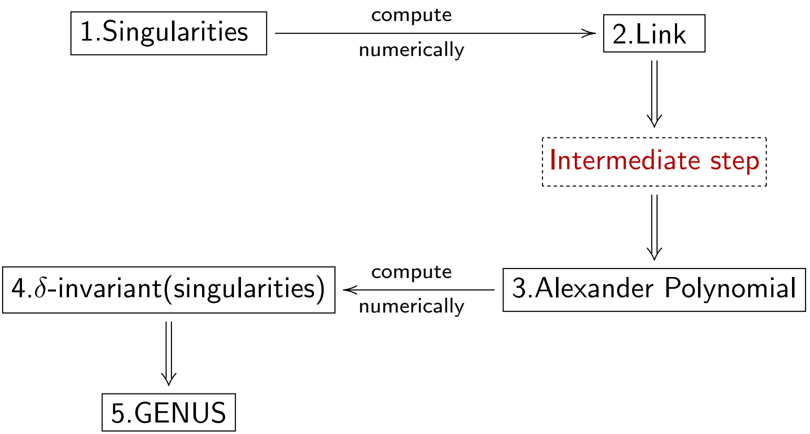
Computing the Alexander polynomial of the link

- Input:
  - $L = K_1 \cup \ldots \cup K_m$ with n - crossings
  - $D(L)$- oriented diagram of L
- Output:
  - $\Delta_L(t_1, \ldots t_m) \in \mathbb{Z}[t_1^{\pm 1}, \ldots, t_m^{\pm 1}]$
- Method: consists of several steps



- Need D(L)!

Next


1.Singularities
compute
numerically
2.Link
Intermediate step
3.Alexander Polynomial
compute
numerically
4.δ-invariant(singularities)
5.GENUS

Intermediate step

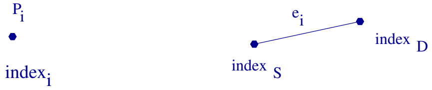
- Input:
  - $Graph(L) = \langle \mathcal{V}, \mathcal{E} \rangle$ with
    $\mathcal{V} = \{p = (m, n, q) \in \mathbb{R}^3\}$
    $\mathcal{E} = \{(i, j) | i, j \in \mathcal{V}\}$
- Output:
  - $D(L)$ s.t.
    $D(L)$ is the image under regular projection of L together with the information on each crossing telling which branch goes under and which goes over.

Intermediate step

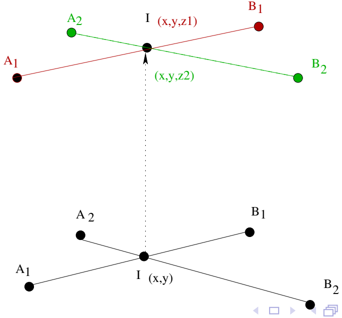
- Input:
  - $Graph'(L) = \langle \mathcal{V}', \mathcal{E}' \rangle$ with
    $\mathcal{V}' = \{p = (m, n) \in \mathbb{R}^2\}$
    $\mathcal{E}' = \{(i, j) | i, j \in \mathcal{V}'\}$
- Output:
  - $D(L)$ s.t. $D(L)$ is the image under regular projection of L together with the information on each crossing telling which branch goes under and which goes over.
- Method: the Bentley-Ottman algorithm

Intermediate step

- Input:
  - $E = \{e_1, e_2, ..., e_n\}$-set of n edges in the plane with:

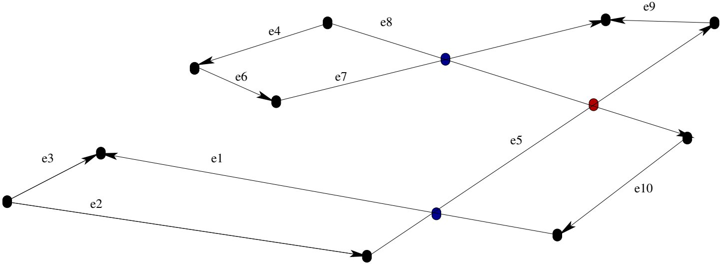


- Output:
  - $I = \{p_i = (a_i, b_i) \in \mathbb{R}^2 \mid \forall p_i \; \exists e_j \neq e_k : e_j \cap e_k = \{p_i\}\}$
    $E_I = \{(e_i, e_j) \mid i \neq j, e_i \cap e_j \neq \emptyset, z(e_i) < z(e_j))\}$



Intermediate step

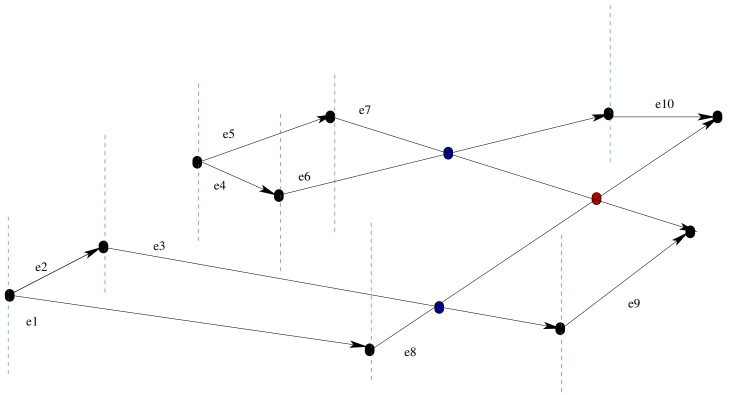
Given:



$$E = \{e_2, e_3, e_1, e_6, e_4, e_7, e_8, e_5, e_9, e_{10}\}$$

Intermediate step

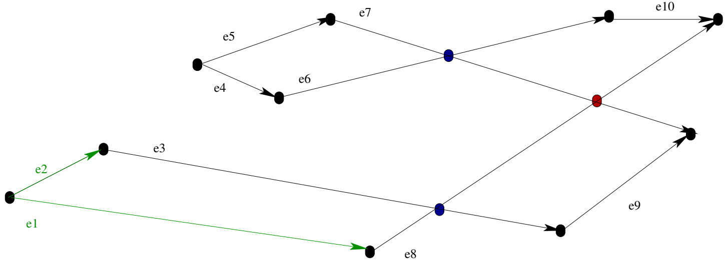
Sorting the edges-necessary condition!



$$E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}\}$$

Intermediate step

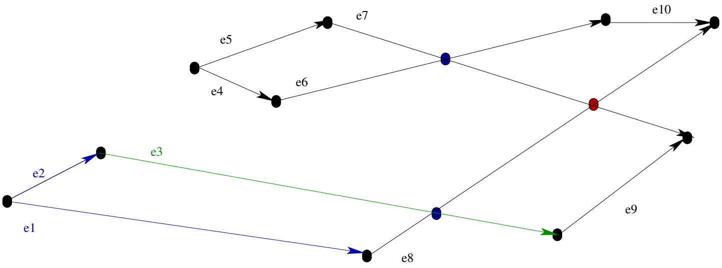
Initialization:



$E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}\}$
$SW_{list} = \{e_1, e_2\}$

Intermediate step

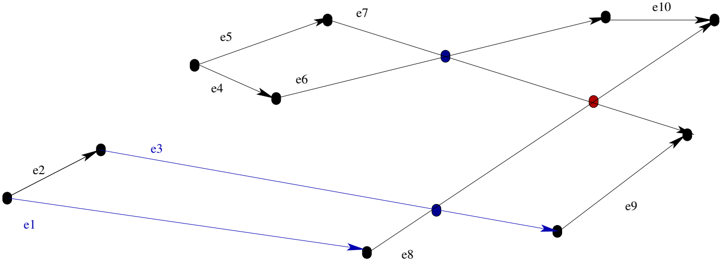
Step 1:



$E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}\}$
$SW_{list} = \{e_1, e_2\}$

Intermediate step

Step 1:



$E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}\}$

$SW_{list} = \{e_1, e_3\}$

Test $e_3 \cap e_1$? No!

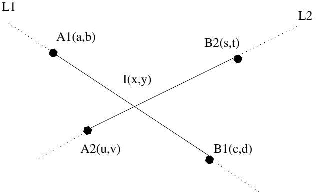
$I = \emptyset$

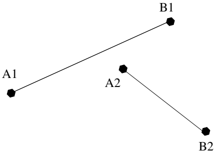
$E_I = \emptyset$

Doctoral Program Computational Mathematics Numerical Analysis and Symbolic Computation

Intermediate step

How do we test intersection of 2 edges?

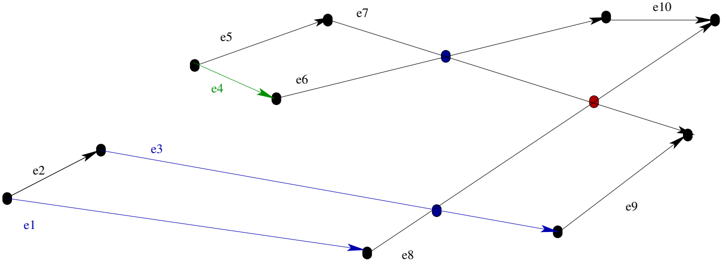




- $L_1(x,y) : y_1 = m_1 \cdot x + b_1$
  $m_1 = \dfrac{d-b}{c-a}, b_1 = \dfrac{b \cdot c - a \cdot d}{c-a}$
  $L_1(x,y) : (b-d)x + (c-a)y + a \cdot d - b \cdot c = 0$
  $L_2(x,y) : (v-t)x + (s-u)y + u \cdot t - v \cdot s = 0$
  (similarly to L_1)
- Note:
  - $L_1(A_2) \cdot L_1(B_2) < 0 \Rightarrow e_1 \cap e_2 \neq \emptyset$
    $L_2(A_1) \cdot L_2(B_1) < 0 \Rightarrow e_1 \cap e_2 \neq \emptyset$
  - $L_1(A_2) \cdot L_1(B_2) > 0 \Rightarrow e_1 \cap e_2 = \emptyset$
    $L_2(A_1) \cdot L_2(B_1) > 0 \Rightarrow e_1 \cap e_2 = \emptyset$
  - $L_1(A_2) \cdot L_1(B_2) = 0 \Rightarrow e_1 \cap e_2 = \{A_2\}$,
    $L_2(A_1) \cdot L_2(B_1) = 0 \Rightarrow e_1 \cap e_2 = \{A_2\}$,
    $(A_2 = B_1)$

Intermediate step

Step 2:

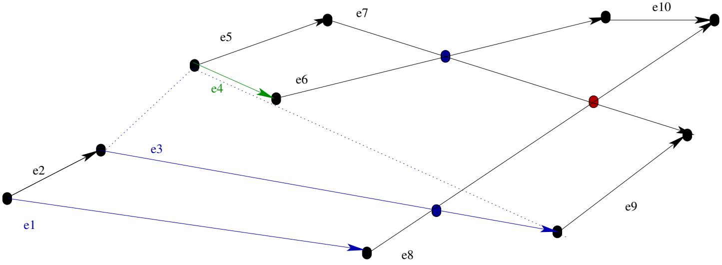


$$E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}\}$$
$$SW_{list} = \{e_1, e_3\}$$

Doctoral Program Computational Mathematics Numerical Analysis and Symbolic Computation

Intermediate step

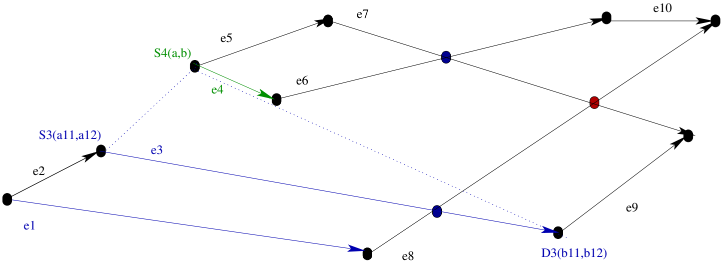
Step 2:



$E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}\}$
$SW_{list} = \{e_1, e_3\}$

Intermediate step

Step 2:



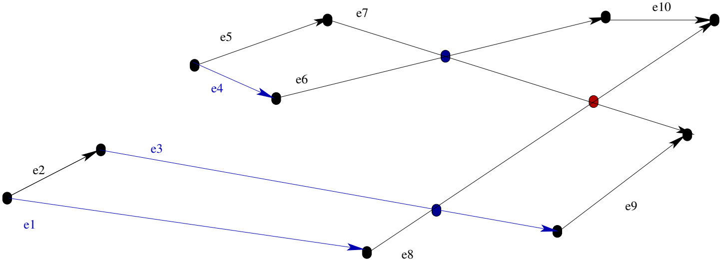
$E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}\}$
$SW_{list} = \{e_1, e_3\}$; compute:

$$det(e_4, e_3) = \begin{pmatrix} a_{11} & a_{12} & 1 \\ b_{11} & b_{12} & 1 \\ a & b & 1 \end{pmatrix} > 0 \Rightarrow e_4 \text{ after } e_3 \text{ in } SW_{list}$$

Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

Intermediate step

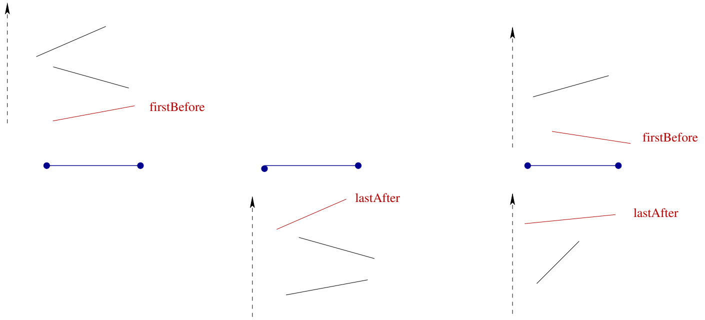
Step 2:



$E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}\}$
$SW_{list} = \{e_1, e_3, e_4\}$
Test $e_4 \cap e_3$? No!
$I = \emptyset$
$E_I = \emptyset$

Intermediate step

Keeping the sweep events list ordered - necessary condition!

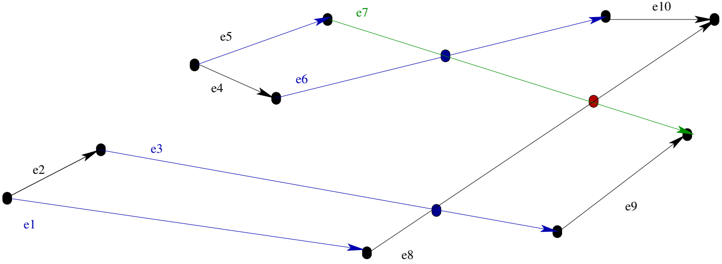


Given e, if $\forall e_i : det(e, e_i) < 0 \Rightarrow e$ before e_i
Given e, if $\forall e_i : det(e, e_i) > 0 \Rightarrow e$ after e_i

Intermediate step

....

Step 5:

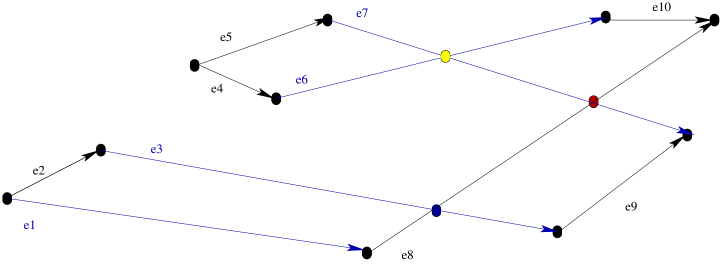


$E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}\}$

$SW_{list} = \{e_1, e_3, e_6, e_5\}$

Intermediate step

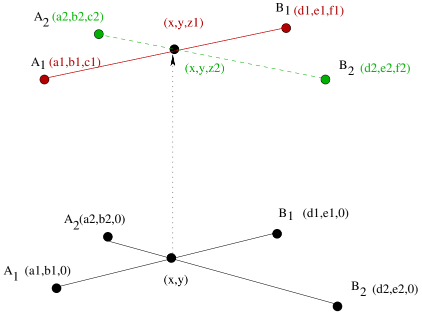
Step 5:



$E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}\}$
$SW_{list} = \{e_1, e_3, e_6, e_7\}$
Test $e_7 \cap e_6$? Yes! $\Rightarrow$
$SW_{list} = \{e_1, e_3, e_7, e_6\}$
$I = \{(a_1, b_1)\}$,
$E_I = \{(e_6, e_7)\}$

Intermediate step

Managing the info on each crossing in $\mathbb{R}^3$:



For:

$$L_2(A_1) = \begin{pmatrix} a_2 & b_2 & 1 \\ d_2 & e_2 & 1 \\ a_1 & b_1 & 1 \end{pmatrix}$$

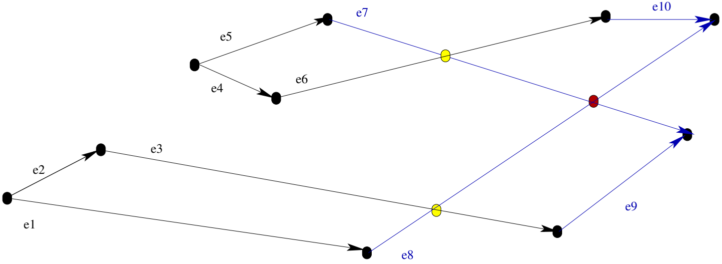
$$L_2(B_1) = \begin{pmatrix} a_2 & b_2 & 1 \\ d_2 & e_2 & 1 \\ d_1 & e_1 & 1 \end{pmatrix}$$

- Compute α_1 from:
  $\alpha_1 \cdot L_2(A_1) + (1 - \alpha_1) \cdot L_2(B_1) = 0$,
  Compute z_1 from:
  $z_1 = \alpha_1 \cdot c_1 + (1 - \alpha_1) \cdot f_1$
  (similarly compute z_2)
- Note: compare $z_1 ? z_2$
  - $z_1 > z_2 \Rightarrow e_1$ over e_2
  - $z_1 < z_2 \Rightarrow e_1$ under e_2

Intermediate step

....

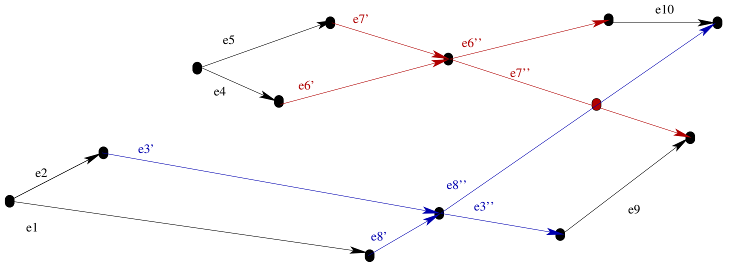
Final step:



$E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}\}$
$SW_{list} = \{e_9, e_8, e_7, e_{10}\}$
$I = \{(a_1, b_1), (a_2, b_2)\}$
$E_I = \{(e_6, e_7), (e_8, e_3)\}$

Intermediate step

Refinements of the algorithm:



Everytime an intersection is detected we update E, SW_{list} as follows:

Detect $e_6 \cap e_7$:

$E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_7^{''}, e_6^{''}, e_9, e_{10}\}$

$SW_{list} = \{e_1, e_3, e_7^{'}, e_6^{'}\}$

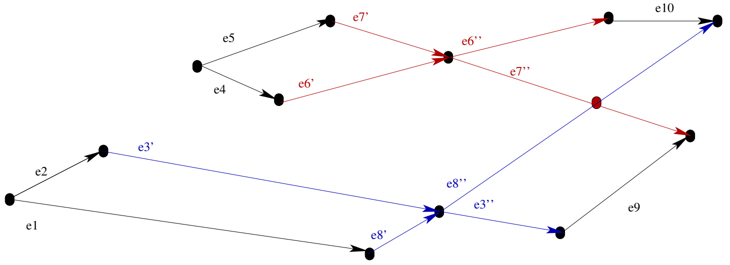
Detect $e_8 \cap e_3$:

$E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_3^{''}, e_8^{''}, e_7^{''}, e_6^{''}, e_9, e_{10}\}$

$SW_{list} = \{e_3^{'}, e_8^{'}, e_7^{'}, e_6^{'}\}$

Intermediate step

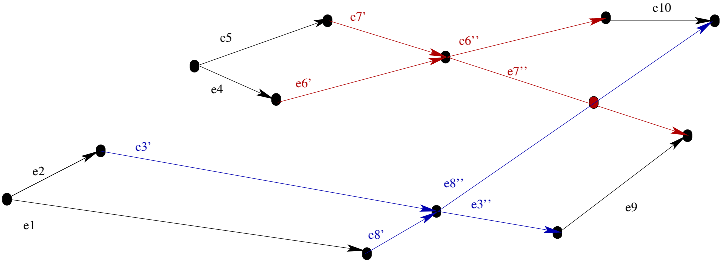
Refinements of the algorithm (next):



$SW_{list} = \{e_1, e_3, e_7^{'}, e_6^{'}\}$
$E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_7^{''}, e_6^{''}, e_9, e_{10}\}$
$SW_{list} = \{e_8, e_3, e_7^{'}, e_6^{'}\}$
$E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_3^{''}, e_8^{''}, e_7^{''}, e_6^{''}, e_9, e_{10}\}$
$SW_{list} = \{e_3^{'}, e_8^{'}, e_7^{'}, e_6^{'}\}$

Intermediate step

Refinements of the algorithm (next):



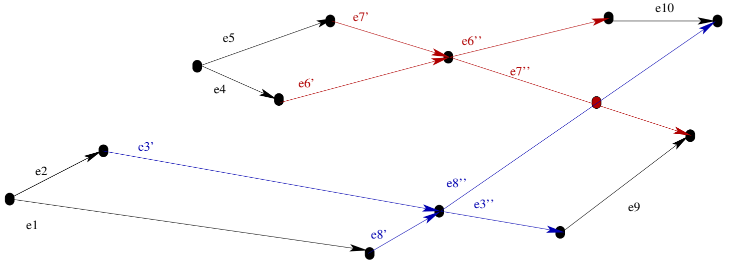
$SW_{list} = \{e_3', e_8', e_7', e_6'\}$

$E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_3'', e_8'', e_7'', e_6'', e_9, e_{10}\}$

$SW_{list} = \{e_3'', e_8', e_7', e_6'\}$

Intermediate step

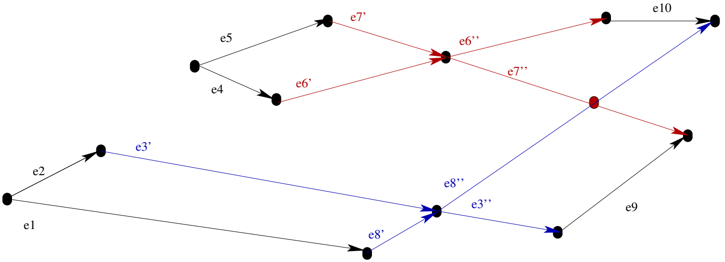
Refinements of the algorithm (next):



$SW_{list} = \{e_3'', e_8', e_7', e_6'\}$
$E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_3'', e_8'', e_7'', e_6'', e_9, e_{10}\}$
$SW_{list} = \{e_3'', , e_8'', e_7', e_6'\}$

Intermediate step

Refinements of the algorithm (next):



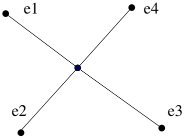
$SW_{list} = \{e_3^{''}, e_8^{'}, e_7^{'}, e_6^{'}\}$
$E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_3^{''}, e_8^{''}, e_7^{''}, e_6^{''}, e_9, e_{10}\}$
$SW_{list} = \{e_3^{''}, , e_8^{''}, e_7^{''}, e_6^{'}\} \Rightarrow$
$I = \{(a_1, b_1), (a_2, b_2), (a_3, b_3)\}$
$E_I = \{(e_6, e_7), (e_8, e_3), (e_8^{''}, e_7^{''})\}$

Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

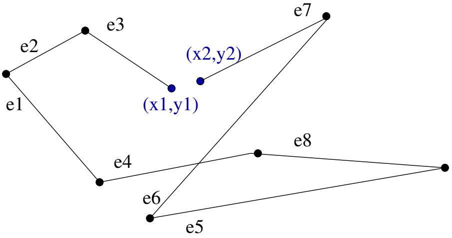
Intermediate step

Degenerate cases in the algorithm (unsolved yet!):

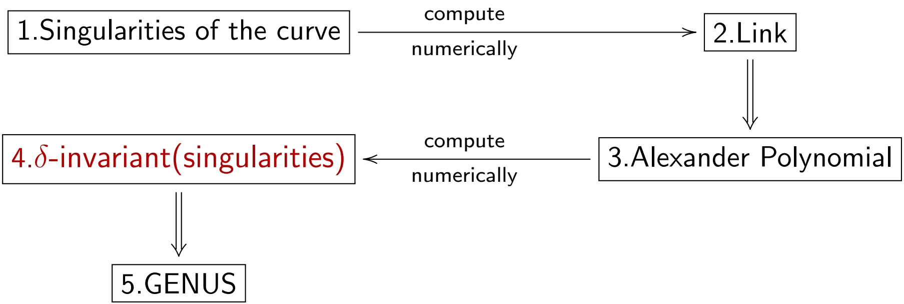
Case 1: Find condition s.t. $I = e_1 \cap e_2 \cap e_3 \cap e4$



Case 2: Find $\epsilon > 0$ s.t. $(x1 - x2)^2 + (y1 - y2)^2 < \epsilon^2$



Next


1.Singularities of the curve
compute
numerically
2.Link
3.Alexander Polynomial
compute
numerically
4.δ-invariant(singularities)
5.GENUS

Computing the δ-invariant of the singularity

- Input:
  - $C \subset \mathbb{C}^2$ complex algebraic curve;
  - $z \in Singularities(C)$;
  - $\Delta(t_1, .., t_p)$ - Alexander polynomial of z;
  - $r =$ number of variables in Δ (branches of C through z);
  - $\mu =$ degree of Δ (multiplicity of z);
- Output:
  - $\delta_z > 0$ s.t.

δ_z is an invariant that measures
the number of double points of C at z.

Computing the δ-invariant of the singularity

- Method: based on Milnor's research on singularities of complex hypersurfaces.

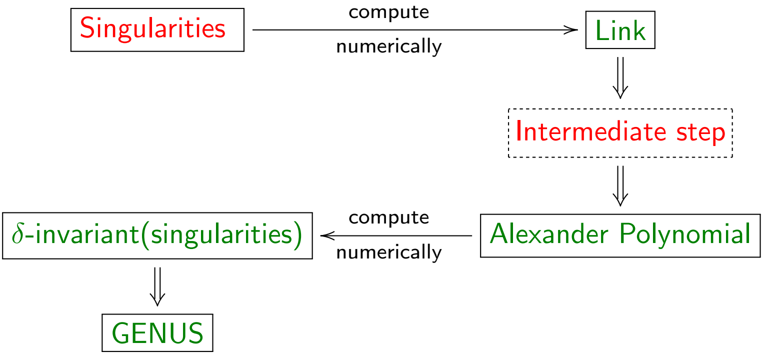
$$\Delta(t_1, ..., t_p) : r, \mu$$

$$r \geq 2 \Rightarrow \delta = \frac{1}{2}(\mu + r)$$

$$r = 1 \Rightarrow \delta = \frac{1}{2}\mu$$

Summary

- At present: for symbolic coefficients



- Future work: tests for algorithm with numeric coefficients

Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

Conclusion

- first results and test experiments were presented;
- Future work:
  - deeper introspection into some mathematical aspects (i.e. Milnor's fibration, Alexander polynomial);
  - correctness/completeness for the algorithm;
  - implementation of the algorithm;
  - analysis of the algorithm.

Thank you for your attention.