Symbolic numeric algorithms for genus computation based on knot theory

Mădălina Hodorog[1]
Supervisor: Josef Schicho[1]
Joint work with Bernard Mourrain[2]

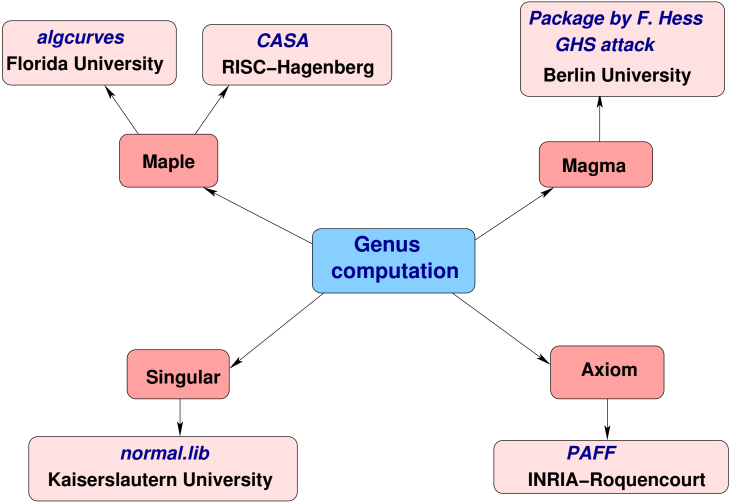
[1]Johann Radon Institute for Computational and Applied Mathematics,
Austrian Academy of Sciences,
Research Institute for Symbolic Computation
Johannes Kepler University Linz, Austria
[2]INRIA Sophia-Antipolis, France

January 8, 2010

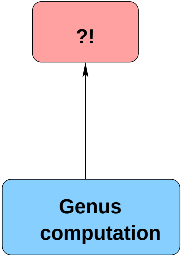
Table of contents

Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

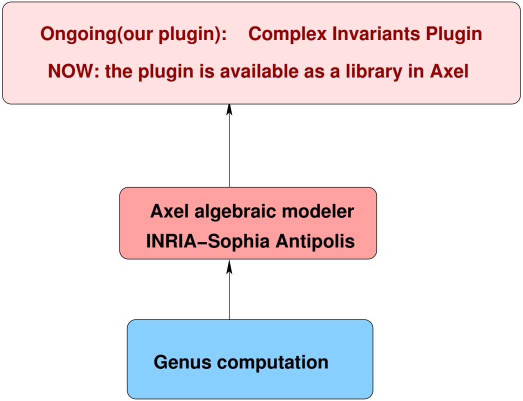
Symbolic Algorithms:



Numeric Algorithms:



A proposal for a symbolic-numeric algorithm: DK9 Project: Symbolic-Numeric techniques for genus computation and parametrization (initiated by Prof. Josef Schicho).



Another proposal: Recently, another numeric method (different from ours) for genus computation was reported.

What?

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

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

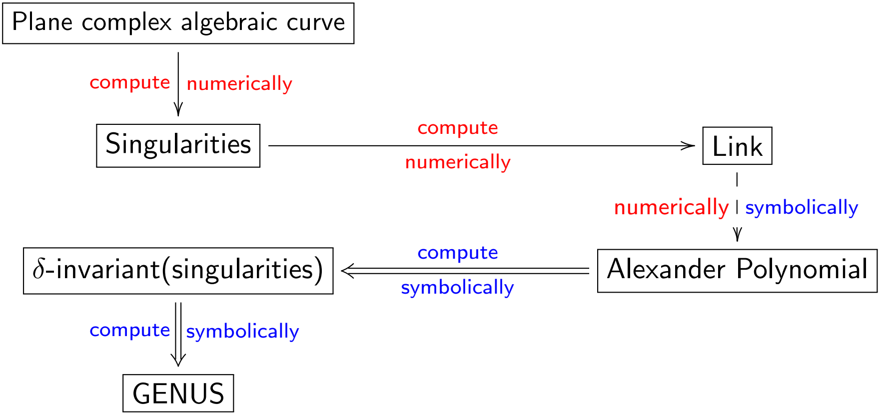
where $Sing(C)$ is the set of singularities of the curve C.

Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

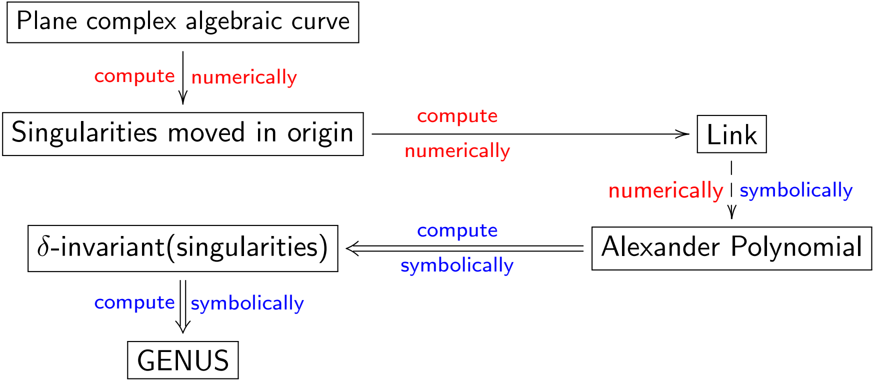
How?

- Strategy for computing the genus



How?

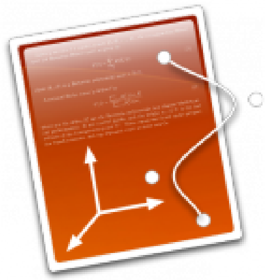
- Strategy for computing the genus



Solving the problem

Implementation of the algorithm

- *Axel* algebraic geometric modeler [a]
  - developed by *Galaad* team (INRIA Sophia-Antipolis);
  - written in $C{+}{+}$, Qt Script for Applications (QSA);
  - provides algebraic tools for:
    - implicit surfaces;
    - implicit curves.
  - free, available at:



[a] Acknowledgements: Julien Wintz

Solving the problem

Implementation of the algorithm

- *Axel* algebraic geometric modeler [a]
  - developed by *Galaad* team (INRIA Sophia-Antipolis);
  - written in $C++$, Qt Script for Applications (QSA);
  - provides algebraic tools for:
    - implicit surfaces;
    - implicit curves.
  - free, available at:

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE — INRIA

[a] Acknowledgements: Julien Wintz

Solving the problem

Implementation of the algorithm

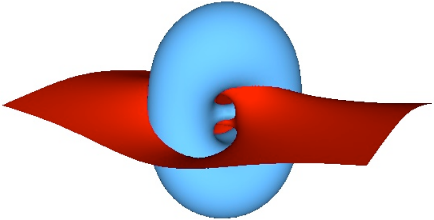
- *Axel* algebraic geometric modeler [a]
  - developed by *Galaad* team (INRIA Sophia-Antipolis);
  - written in *C++*, Qt Script for Applications (QSA);
  - provides algebraic tools for:
    - implicit surfaces;
    - implicit curves.
  - free, available at:

[a] Acknowledgements: Julien Wintz

Solving the problem

Implementation of the algorithm

- *Axel* algebraic geometric modeler [a]
  - developed by *Galaad* team (INRIA Sophia-Antipolis);
  - written in *C++*, Qt Script for Applications (QSA);
  - provides algebraic tools for:
    - implicit surfaces;
    - implicit curves.
  - free, available at:

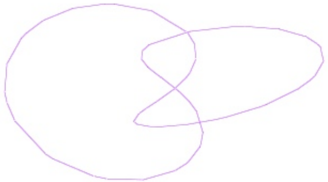


[a] Acknowledgements: Julien Wintz

Solving the problem

Implementation of the algorithm

- *Axel* algebraic geometric modeler [a]
  - developed by *Galaad* team (INRIA Sophia-Antipolis);
  - written in *C++*, Qt Script for Applications (QSA);
  - provides algebraic tools for:
    - implicit surfaces;
    - implicit curves.
  - free, available at:



[a] Acknowledgements: Julien Wintz

Solving the problem

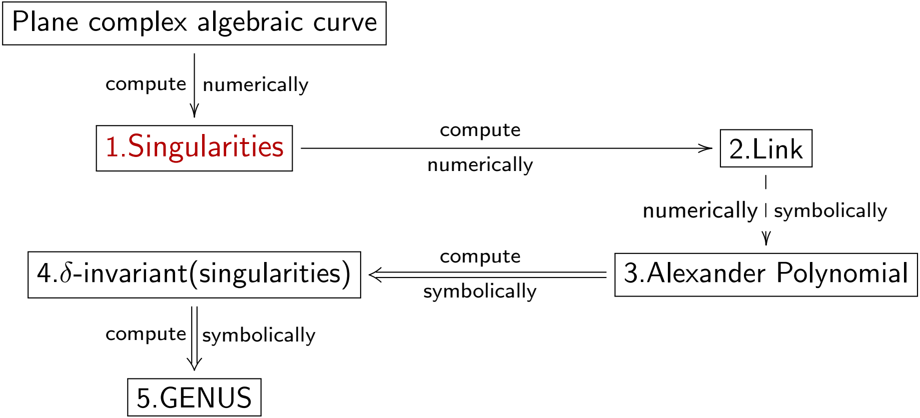
Implementation of the algorithm

- *Axel* algebraic geometric modeler [a]
  - developed by *Galaad* team (INRIA Sophia-Antipolis);
  - written in *C++*, Qt Script for Applications (QSA);
  - provides algebraic tools for:
    - implicit surfaces;
    - implicit curves.
  - free, available at:

http://axel.inria.fr/

[a] Acknowledgements: Julien Wintz

First


Plane complex algebraic curve
compute
numerically
1.Singularities
compute
numerically
2.Link
numerically
symbolically
3.Alexander Polynomial
compute
symbolically
4.δ-invariant(singularities)
compute
symbolically
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:
  - $Sing(C) = \{(z_0,w_0) \in \mathbb{C}^2 | F(z_0,w_0) = 0, \dfrac{\partial F}{\partial z}(z_0,w_0) = 0, \dfrac{\partial F}{\partial w}(z_0,w_0) = 0\}$

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

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

Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

Computing the singularities of the curve

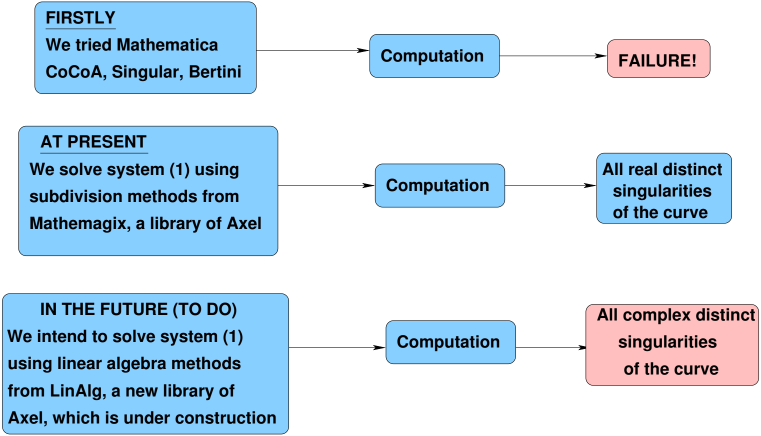
or in $\mathbb{R}^4$: $F(z,w) = F(x+iy, u+iv) = s(x,y,u,v) + it(x,y,u,v)$

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

Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

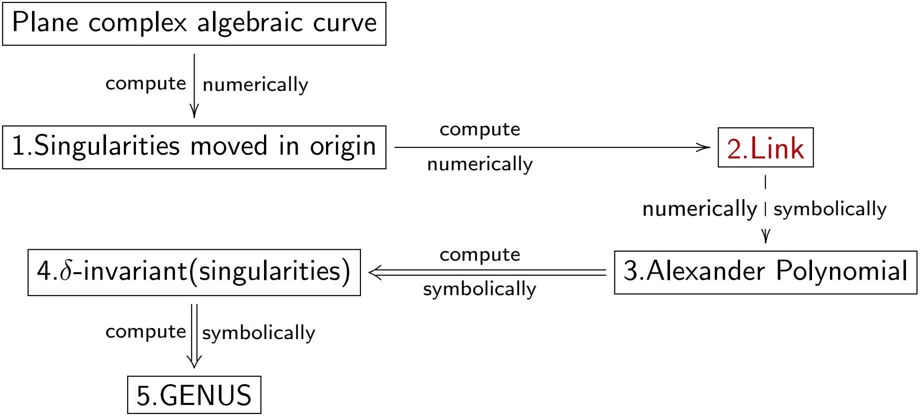
Computing the singularities of the curve

For input polynomials with numeric coefficients



Note: so far this is an open problem.

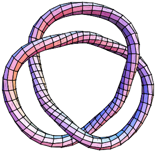
Next


Plane complex algebraic curve
compute
numerically
1.Singularities moved in origin
compute
numerically
2.Link
numerically
symbolically
3.Alexander Polynomial
compute
symbolically
4.δ-invariant(singularities)
compute
symbolically
5.GENUS

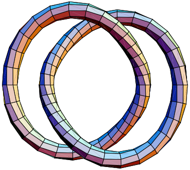
Knot theory - preliminaries

- A **knot** is a simple closed curve in $\mathbb{R}^3$.
- A **link** is a finite union of disjoint knots.
- Links resulted from the intersection of a given curve with the sphere are called **algebraic links**.
  Note: Alexander polynomial is a complete invariant for the algebraic links (Yamamoto 1984).

Trefoil Knot

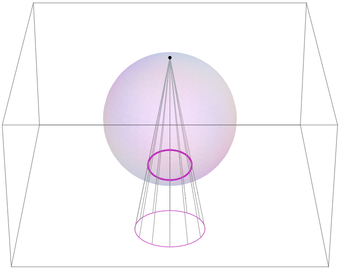


Hopf Link



Computing the link of the singularity

- Why the link of a singularity?
  - helps to study 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 $C = \{(x,y,u,v) \in \mathbb{R}^4 | F(x,y,u,v) = 0\}$ s.t. $(0,0,0,0) \in Sing(C)$

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

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

4. Compute $f(X) = \{(a,b,c,) \in \mathbb{R}^3 | F(...) = 0\} \Leftrightarrow$
   $f(X) = \{(a,b,c) \in \mathbb{R}^3 | ReF(...) = 0, ImF(...) = 0\}$
   For small ϵ, $f(X)$ is a link.

Computing the link of the singularity

Why Axel?

It computes numerically the topology of smooth implicit curves in $\mathbb{R}^3$

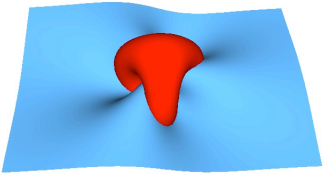
- For $C^4 = \{(z,w) \in \mathbb{C}^2 | z^3 - w^2 = 0\} \subset \mathbb{R}^4$ get
- $f(C^4 \cap S) := C =$
  $= \{(a,b,c) \in \mathbb{R}^3 | ReF(...) = 0, ImF(...) = 0\}$
- compute $Graph(C) = \langle \mathcal{V}, \mathcal{E} \rangle$ with
  $\mathcal{V} = \{p = (m,n,q) \in \mathbb{R}^3\}$
  $\mathcal{E} = \{(i,j) | i,j \in \mathcal{V}\}$
- s.t. $Graph(C) \cong_{isotopic} C$
- we also compute
  $S^{'} = \{(a,b,c) \in \mathbb{R}^3 | ReF(...) + ImF(...) = 0\}$
  $S^{''} = \{(a,b,c) \in \mathbb{R}^3 | Re(F) - ImF(...) = 0\}$
- C is at the intersection of any of the two surfaces
  $ReF(...), ImF(...)$
  $ReF(...) + ImF(...), ReF(...) - ImF(...)$

Computing the link of the singularity

Why Axel?

It computes numerically the topology of smooth implicit curves in $\mathbb{R}^3$

- For $C^4 = \{(z,w) \in \mathbb{C}^2 | z^3 - w^2 = 0\} \subset \mathbb{R}^4$ get
- $f(C^4 \cap S) := C =$
  $= \{(a,b,c) \in \mathbb{R}^3 | ReF(...) = 0, ImF(...) = 0\}$
- compute $Graph(C) = \langle \mathcal{V}, \mathcal{E} \rangle$ with
  $\mathcal{V} = \{p = (m,n,q) \in \mathbb{R}^3\}$
  $\mathcal{E} = \{(i,j) | i,j \in \mathcal{V}\}$
- s.t. $Graph(C) \cong_{isotopic} C$
- we also compute
  $S' = \{(a,b,c) \in \mathbb{R}^3 | ReF(...) + ImF(...) = 0\}$
  $S'' = \{(a,b,c) \in \mathbb{R}^3 | Re(F) - ImF(...) = 0\}$
- C is at the intersection of any of the two surfaces
  $ReF(...), ImF(...)$
  $ReF(...) + ImF(...), ReF(...) - ImF(...)$



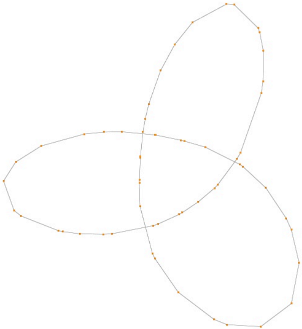
Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

Computing the link of the singularity

Why Axel?

It computes numerically the topology of smooth implicit curves in $\mathbb{R}^3$

- For $C^4 = \{(z,w) \in \mathbb{C}^2 | z^3 - w^2 = 0\} \subset \mathbb{R}^4$ get
- $f(C^4 \cap S) := C =$
  $= \{(a,b,c) \in \mathbb{R}^3 | ReF(...) = 0, ImF(...) = 0\}$
- compute $Graph(C) = \langle \mathcal{V}, \mathcal{E} \rangle$ with
  $\mathcal{V} = \{p = (m,n,q) \in \mathbb{R}^3\}$
  $\mathcal{E} = \{(i,j) | i,j \in \mathcal{V}\}$
- s.t. $Graph(C) \cong_{isotopic} C$
- we also compute
  $S^{'} = \{(a,b,c) \in \mathbb{R}^3 | ReF(...) + ImF(...) = 0\}$
  $S^{''} = \{(a,b,c) \in \mathbb{R}^3 | Re(F) - ImF(...) = 0\}$
- C is at the intersection of any of the two surfaces
  $ReF(...), ImF(...)$
  $ReF(...) + ImF(...), ReF(...) - ImF(...)$



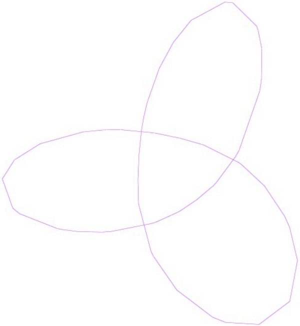
Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

Computing the link of the singularity

Why Axel?

It computes numerically the topology of smooth implicit curves in $\mathbb{R}^3$

- For $C^4 = \{(z, w) \in \mathbb{C}^2 | z^3 - w^2 = 0\} \subset \mathbb{R}^4$ get
- $f(C^4 \cap S) := C =$
  $= \{(a, b, c) \in \mathbb{R}^3 | ReF(...) = 0, ImF(...) = 0\}$
- compute $Graph(C) = \langle \mathcal{V}, \mathcal{E} \rangle$ with
  $\mathcal{V} = \{p = (m, n, q) \in \mathbb{R}^3\}$
  $\mathcal{E} = \{(i, j) | i, j \in \mathcal{V}\}$
- s.t. $Graph(C) \cong_{isotopic} C$
- we also compute
  $S^{'} = \{(a, b, c) \in \mathbb{R}^3 | ReF(...) + ImF(...) = 0\}$
  $S^{''} = \{(a, b, c) \in \mathbb{R}^3 | Re(F) - ImF(...) = 0\}$
- C is at the intersection of any of the two surfaces
  $ReF(...), ImF(...)$
  $ReF(...) + ImF(...), ReF(...) - ImF(...)$

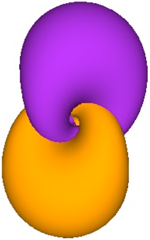


Computing the link of the singularity

Why Axel?

It computes numerically the topology of smooth implicit curves in $\mathbb{R}^3$

- For $C^4 = \{(z,w) \in \mathbb{C}^2 | z^3 - w^2 = 0\} \subset \mathbb{R}^4$ get
- $f(C^4 \cap S) := C =$
  $= \{(a,b,c) \in \mathbb{R}^3 | ReF(...) = 0, ImF(...) = 0\}$
- compute $Graph(C) = \langle \mathcal{V}, \mathcal{E} \rangle$ with
  $\mathcal{V} = \{p = (m,n,q) \in \mathbb{R}^3\}$
  $\mathcal{E} = \{(i,j) | i,j \in \mathcal{V}\}$
- s.t. $Graph(C) \cong_{isotopic} C$
- we also compute
  $S^{'} = \{(a,b,c) \in \mathbb{R}^3 | ReF(...) + ImF(...) = 0\}$
  $S^{''} = \{(a,b,c) \in \mathbb{R}^3 | Re(F) - ImF(...) = 0\}$
- C is at the intersection of any of the two surfaces
  $ReF(...), ImF(...)$
  $ReF(...) + ImF(...), ReF(...) - ImF(...)$

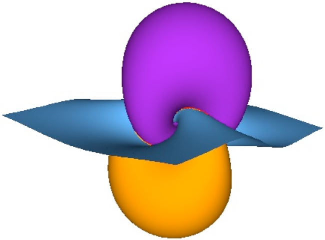


Computing the link of the singularity

Why Axel?

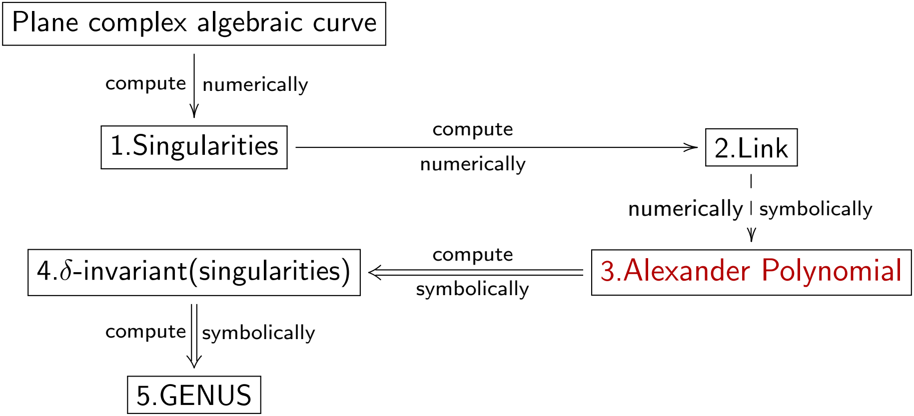
It computes numerically the topology of smooth implicit curves in $\mathbb{R}^3$

- For $C^4 = \{(z,w) \in \mathbb{C}^2 | z^3 - w^2 = 0\} \subset \mathbb{R}^4$ get
- $f(C^4 \cap S) := C =$
  $= \{(a,b,c) \in \mathbb{R}^3 | ReF(...) = 0, ImF(...) = 0\}$
- compute $Graph(C) = \langle \mathcal{V}, \mathcal{E} \rangle$ with
  $\mathcal{V} = \{p = (m,n,q) \in \mathbb{R}^3\}$
  $\mathcal{E} = \{(i,j) | i,j \in \mathcal{V}\}$
- s.t. $Graph(C) \cong_{isotopic} C$
- we also compute
  $S' = \{(a,b,c) \in \mathbb{R}^3 | ReF(...) + ImF(...) = 0\}$
  $S'' = \{(a,b,c) \in \mathbb{R}^3 | Re(F) - ImF(...) = 0\}$
- C is at the intersection of any of the two surfaces
  $ReF(...), ImF(...)$
  $ReF(...) + ImF(...), ReF(...) - ImF(...)$



Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

Next

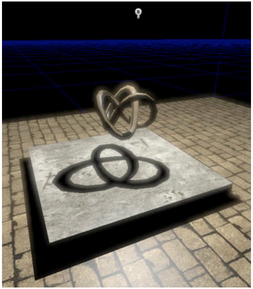


Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

Knot theory - preliminaries

The Alexander polynomial was introduced by J.W. Alexander II in 1928. It depends on the fundamental group of the complement of the knot in $\mathbb{R}^3$.

Definition. Let L be a link with n components. The multivariate Alexander polynomial is a Laurent polynomial $\Delta_L \in \mathbb{Z}[t_0, ..., t_n, t_0^{-1}, ..., t_n^{-1}]$, which is defined up to a factor of $\pm t_0^{k_0} ... t_n^{k_n}, k_i \in \mathbb{Z}, \forall i \in \{0, ..., n\}$.



Note. At present there is no complete invariant to distinguish links in knot theory. But the Alexander polynomial is a complete invariant for the algebraic links (Yamamoto 1984).

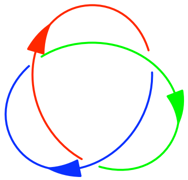
Knot theory - preliminaries

A knot projection is a **regular projection** if no three points on the knot project to the same point, and no vertex projects to the same point as any other point on the knot.

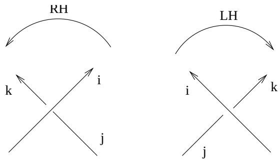
A **diagram** is the image under regular projection, together with the information on each crossing telling which branch goes over and which under.

A crossing is:
-**righthanded** if the underpass traffic goes from right to left.
-**lefthanded** if the underpass traffic goes from left to right.

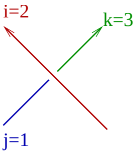
Diagram and arcs

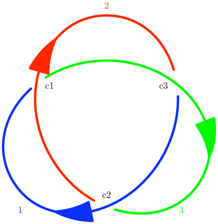


Crossings



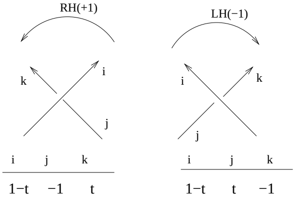
Computing the Alexander polynomial of the link





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

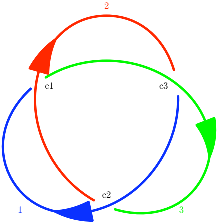
$$P(L) = \left(\quad \quad \right)$$

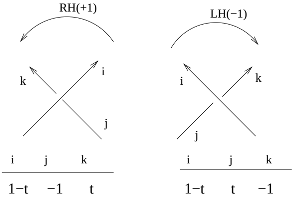


Computing the Alexander polynomial of the link

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

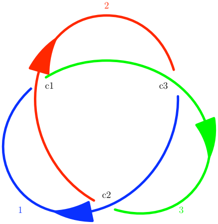
$$P(L) = \left(\begin{array}{ccc} 2 & 1 & 3 \\ 1-t & t & -1 \\ & & \end{array}\right)$$



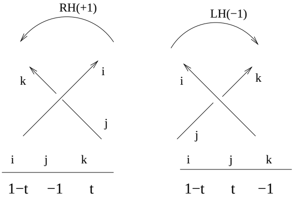


Computing the Alexander polynomial of the link

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



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



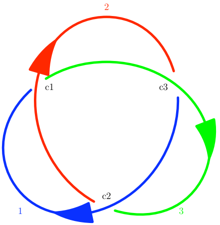
Computing the Alexander polynomial of the link

For a link with $K = 1$ knot:

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

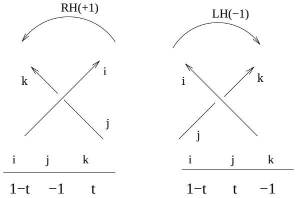
$$D := det(minor(P(L))) = -t^2 + t - 1$$

$$\Delta(L) := \Delta(t) = Normalise(D) = t^2 - t + 1$$



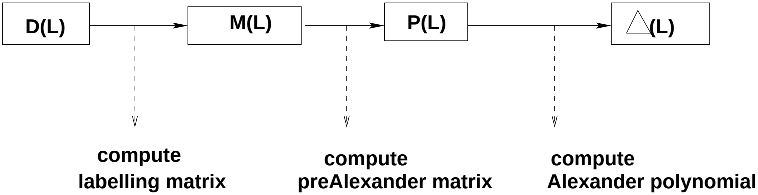
For a link with $K > 1$ knots and n crossings $\Delta(L)$ is the gcd of all the $(n-1) \times (n-1)$ minor determinants of $P(L)$.

Note: The Alexander polynomial is $\Delta(L)$.



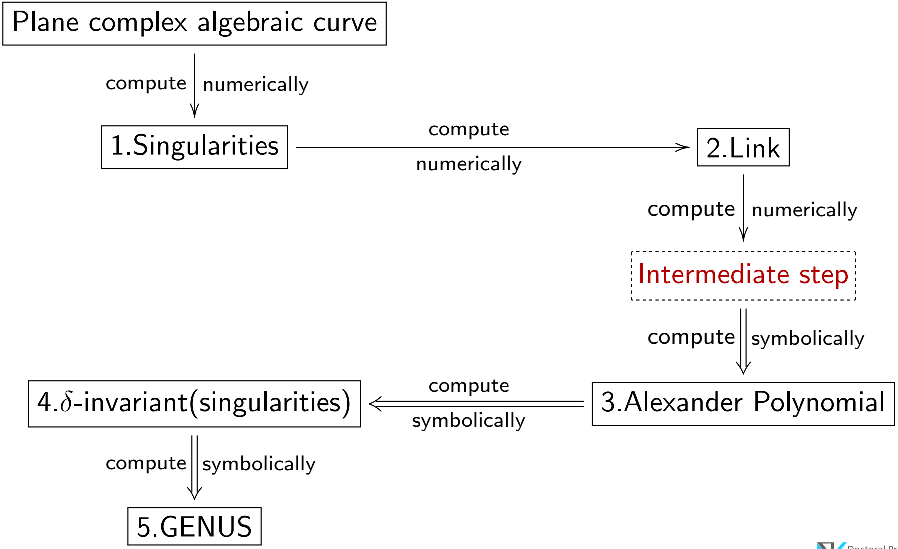
Computing the Alexander polynomial of the link

So, the Alexander polynomial is computed in several steps:

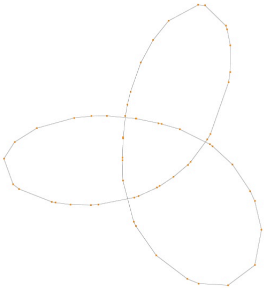


In order to compute it, we need D(L)!

Next


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

Intermediate step

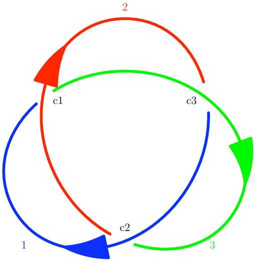
- 
- $G(L) = \langle P, E \rangle$
- p(index,x,y,z)

  e(indexS, indexD)

$\overset{?}{\Rightarrow}$



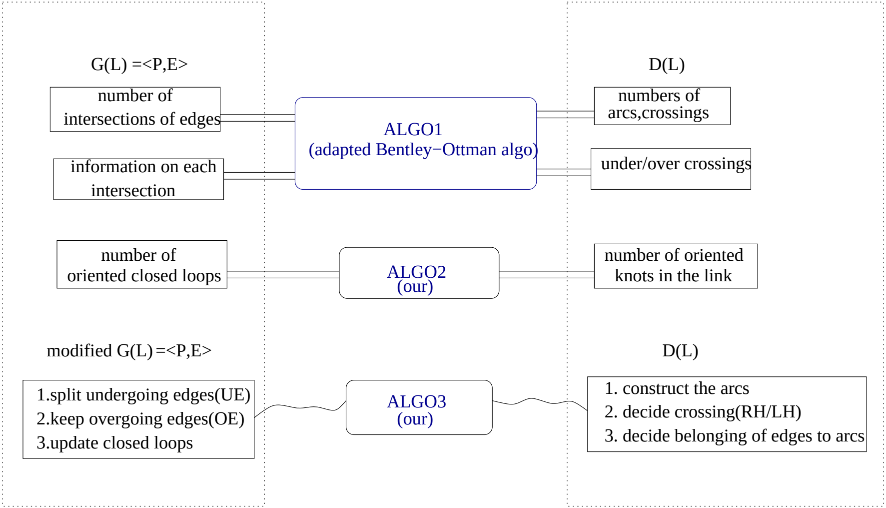
$D(L)$

$\longrightarrow$ number of arcs, crossings

$\longrightarrow$ type of crossings (under, over)

$\longrightarrow$ number of knots in the link(orientation)

Intermediate step

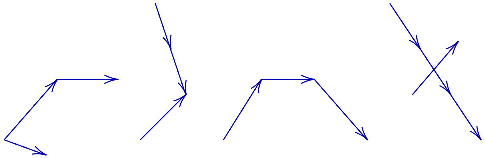


Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

Algorithm 1 - Adapted version of Bentley-Ottman

1

- Input
  - S a set of "short" edges ordered from left to right:

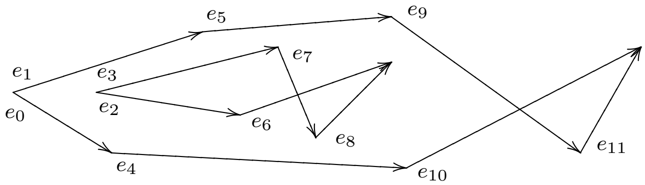


- Output
  - I - the set of all intersections among edges of S and
  - for each $p \in I$, the "arranged" pair of edges (e_i, e_j) such that $p = e_i \cap e_j$.

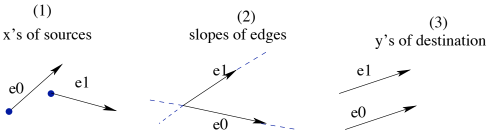
Note: (e_i, e_j) is an "arranged" pair of edges if and only if for $p = e_i \cap e_j$, e_i is below e_j in $\mathbb{R}^3$.

Algorithm 1 - Adapted version of Bentley-Ottman

1



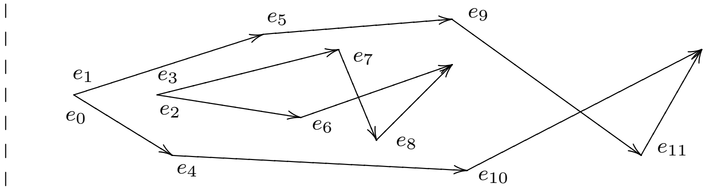
- the edges are ordered by criteria (1),(2),(3):



- the ordering criteria is necessary for the algorithm!

Algorithm 1 - Adapted version of Bentley-Ottman

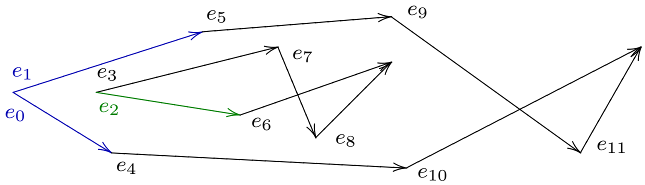
1



- we consider l a sweep line and keep track of two lists:
  $E = \{e_0, e_1, ..., e_{11}\}$ the list of ordered edges
  $Sw = \{?\}$ the list of event points
- while traversing E we insert the edges in Sw in the "right" position:
  - for each $e_i \in E$, we look for an edge $e_j \in Sw$ s.t. source(e_i)=destination(e_j)
  - if such an e_j is not found e_i is inserted in Sw depending on its position against the existing edges in Sw
- That is...

Algorithm 1 - Adapted version of Bentley-Ottman

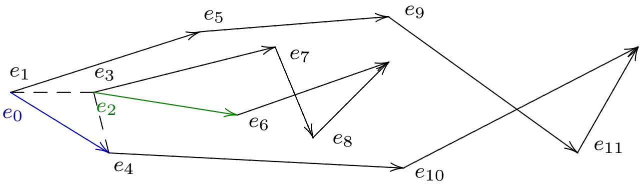
1



- $E = \{e_0, e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}\}$
- $Sw = \{e_0, e_1\}$

Algorithm 1 - Adapted version of Bentley-Ottman

1

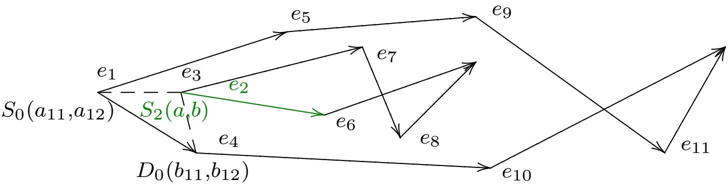


- $E = \{e_0, e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}\}$
- $Sw = \{e_0, e_1\}$

Doctoral Program Computational Mathematics Numerical Analysis and Symbolic Computation

Algorithm 1 - Adapted version of Bentley-Ottman

1

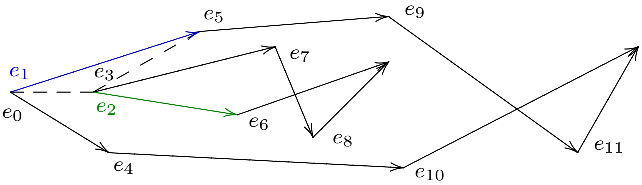


- $E = \{e_0, e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}\}$
- $Sw = \{e_0, e_1\}$; compute:

$$det(e_2, e_0) = \begin{pmatrix} a_{11} & a_{12} & 1 \\ b_{11} & b_{12} & 1 \\ a & b & 1 \end{pmatrix} > 0 \Rightarrow e_2 \text{ after } e_0 \text{ in } Sw$$

Algorithm 1 - Adapted version of Bentley-Ottman

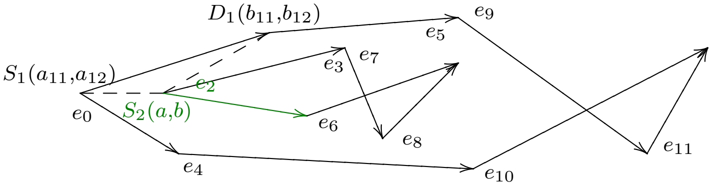
1



- $E = \{e_0, e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}\}$
- $Sw = \{e_0, e_1\}$

Algorithm 1 - Adapted version of Bentley-Ottman

1



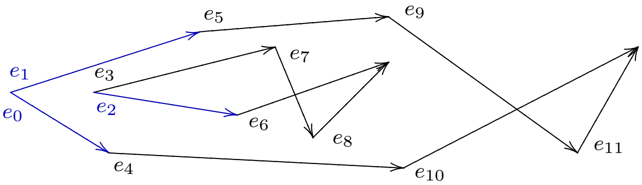
- $E = \{e_0, e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}\}$
- $Sw = \{e_0, e_1\}$; compute:

$$det(e_2, e_1) = \begin{pmatrix} a_{11} & a_{12} & 1 \\ b_{11} & b_{12} & 1 \\ a & b & 1 \end{pmatrix} < 0 \Rightarrow e_2 \text{ before } e_1 \text{ in } Sw$$

Doctoral Program Computational Mathematics Numerical Analysis and Symbolic Computation

Algorithm 1 - Adapted version of Bentley-Ottman

1

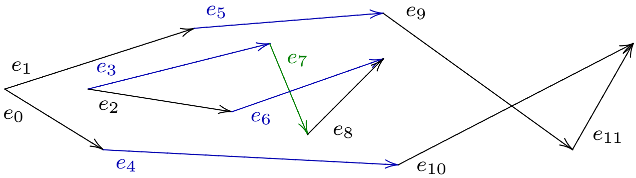


- $E = \{e_0, e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}\}$
- $Sw = \{e_0, e_2, e_1\}$
- Test $e_0 \cap e_2$? No!
  Test $e_2 \cap e_1$? No!
- $I = \emptyset$
  $E_I = \emptyset$

Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

Algorithm 1 - Adapted version of Bentley-Ottman

1

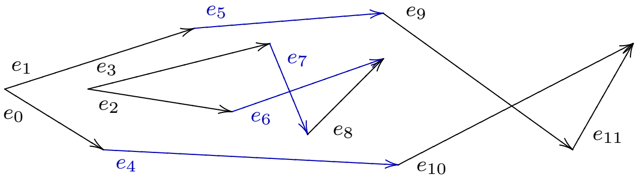


- $E = \{e_0, e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}\}$
- $Sw = \{e_4, e_6, e_3, e_5\}$

Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

Algorithm 1 - Adapted version of Bentley-Ottman

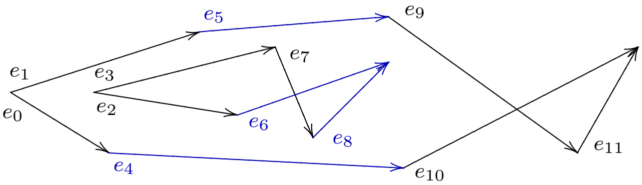
1



- $E = \{e_0, e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}\}$
- $Sw = \{e_4, e_6, e_7, e_5\}$
- Test $e_6 \cap e_7 =?$ Yes!
  Test $e_7 \cap e_5 =?$ No! $\Rightarrow I = \{(x_1, y_1)\}$ $E_I = \{(e_6, e_7)\}$
  $Sw = \{e_4, e_7, e_6, e_5\}$

Algorithm 1 - Adapted version of Bentley-Ottman

1

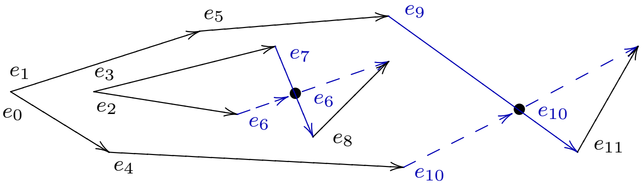


- $E = \{e_0, e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}\}$
- $Sw = \{e_4, e_8, e_6, e_5\}$
- Test $e_4 \cap e_8 =?$No!
  Test $e_8 \cap e_6 =?$No!
- Test $dest(e_4) = dest(e_8)$? No!
  Test $dest(e_8) = dest(e_6)$? Yes! $\Rightarrow$
  $Sw = \{e_4, e_5\}$

Algorithm 1 - Adapted version of Bentley-Ottman

1

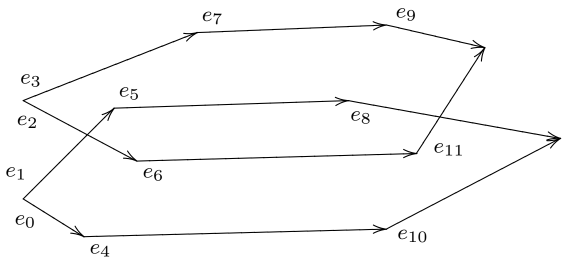
- Final output:



- $I = \{i_1 = (x_1, y_1), i_2 = (x_2, y_2)\}$
  $E_I = \{(e_6, e_7), (e_{10}, e_9)\}$ with
  - e_6 below e_7 in $\mathbb{R}^3$ and
  - e_{10} below e_9 in $\mathbb{R}^3$

Algorithm 2 - Constructing the loops

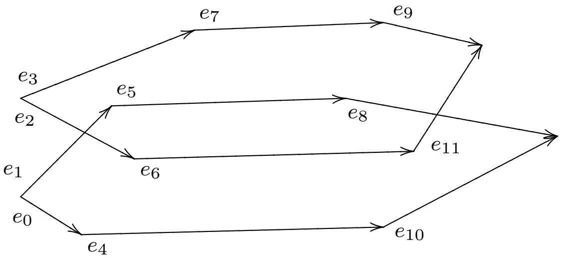
2



- Input
  - E a set of ordered edges by criteria (1),(2),(3)
- Output
  - all the loops $L_{k \in \mathbb{N}} = \{e_{first}, ..., e_{i-1}, e_i, e_{i+1}, .., e_{last}\}$ among E with :
    - for each $e_i \in L$ dest(e_i)=source(e_{i+1})
    - destination(e_{last})=source(e_{first})

Algorithm 2 - Constructing the loops

2

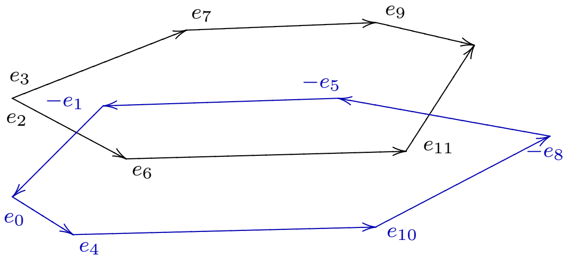


- $E = \{e_0, e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}\}$
- Notation: if e_i=(indexS,indexD) then $-e_i$=(indexD,indexS)
- We apply the following strategy:
  - for each $e_i \in L_k$ we look an edge in E with the same index as dest(e_i)
  - if $e_j \in E$: source(e_j)=dest(e_i) $\Rightarrow L_k = L_k \cup \{e_j\}, E = E \setminus \{e_j\}$
  - if $e_j \in E$: dest(e_j)=dest(e_i) $\Rightarrow L_k = L_k \cup \{-e_j\}, E = E \setminus \{-e_j\}$

Doctoral Program Computational Mathematics
Numerical Analysis and Symbolic Computation

Algorithm 2 - Constructing the loops

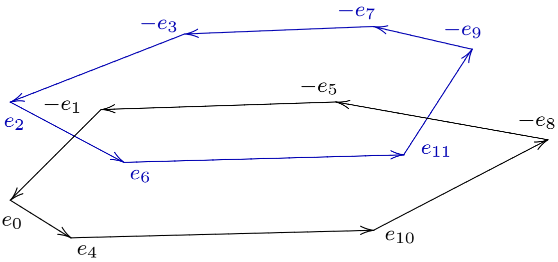
2



- We apply the described strategy for constructing the first loop:
- $E = \{\not{e_0}, \not{e_1}, e_2, e_3, \not{e_4}, \not{e_5}, e_6, e_7, \not{e_8}, e_9, \not{e_{10}}, e_{11}\}$
- $L_0 = \{e_0, e_4, e_{10}, -e_8, -e_5, -e_1\}$

Algorithm 2 - Constructing the loops

2

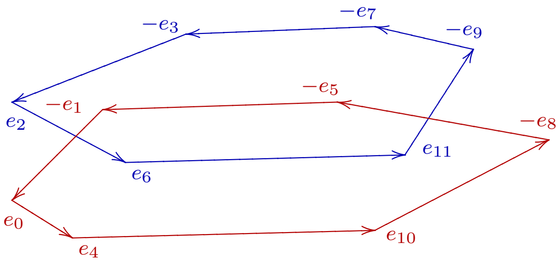


- We apply the same strategy for constructing the next loops until $E = \emptyset$:
- $E = \{\not{e_2}, \not{e_3}, \not{e_6}, \not{e_7}, \not{e_9}, \not{e_{11}}\}$
- $L_1 = \{e_2, e_6, e_{11}, -e_9, -e_7, -e_3\}$
- After this step $E = \emptyset$ so the algorithm terminates.

Algorithm 2 - Constructing the loops

2

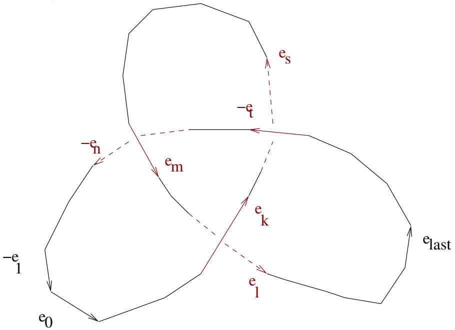
- Final output:



- $L_0 = \{e_0, e_4, e_{10}, -e_8, -e_5, -e_1\}$
  $L_1 = \{e_2, e_6, e_{11}, -e_9, -e_7, -e_3\}$

Algorithm 3 - Constructing the arcs

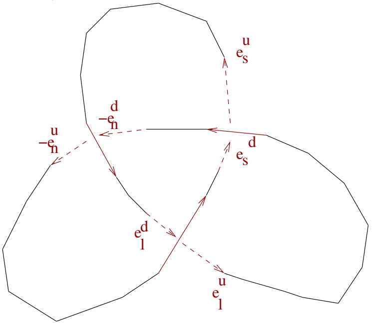
3



- $E = \{e_0, ..., e_n, e_m, ..., e_l, e_k ..., e_t, e_s, ..., e_{last}\}$
  $I = \{(x_1, y_1), (x_2, y_2), (x_3, y_3)\}$
  $E_I = \{(-e_n, e_m), (e_l, e_k), (e_s, -e_t)\}$
- $L_0 = \{e_0, ..., e_k, ..., e_s, ..., e_m, ..., e_l, ..., -e_t, ..., -e_n, ..., -e_1\}$

Algorithm 3 - Constructing the arcs

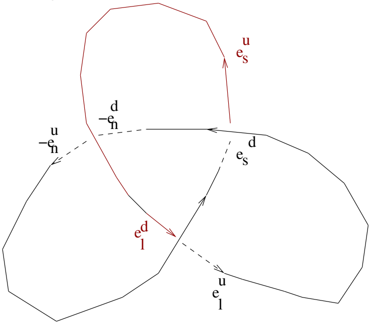
3



- We modify the loops depending on each undergoing edge from $E_I = \{(-e_n, e_m), (e_l, e_k), (e_s, -e_t)\}$
- That is we split all the undergoing edges in two parts.
- $L_0 = \{e_0, ..., e_k, ..., e_s, ..., e_m, ..., e_l, ..., -e_t, ..., -e_n, ..., -e_1\}$ becomes $L_0 = \{e_0, .., e_k, .., e_s^d, e_s^u, .., e_m, .., e_l^d, e_l^u, .., -e_t, .., -e_n^d, -e_n^u, .., -e_1\}$

Algorithm 3 - Constructing the arcs

3

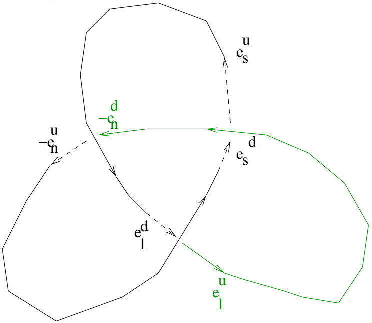


- An arc contains the edges between an edge of type e_i^u and the next consecutive edge of type e_j^d
- From the modified loop we compute the arcs until $L_0 = \emptyset$:

$$L_0 = \{e_0, .., e_k, .., e_s^d, e_s^u, .., e_m, .., e_l^d, e_l^u, .., -e_t, .., -e_n^d, -e_n^u, .., -e_1\}$$
$$L_0 = \{e_0, .., e_k, .., e_s^d, [e_s^u, .., e_m, .., e_l^d], e_l^u, .., -e_t, .., -e_n^d, -e_n^u, .., -e_1\}$$
$$arc_0 = \{e_s^u, ..., e_m, ..., e_l^d\}$$

Algorithm 3 - Constructing the arcs

3



- From the modified loop we compute the arcs until $L_0 = \emptyset$:

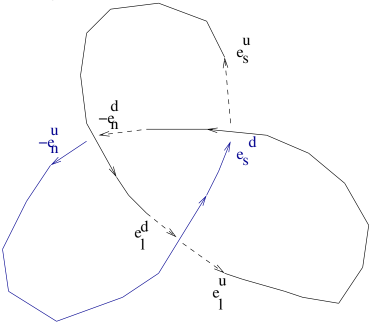
$$L_0 = \{e_0, .., e_k, .., e_s^d, e_l^u, .., -e_t, .., -e_n^d, -e_n^u, .., -e_1\}$$

$$L_0 = \{e_0, .., e_k, .., e_s^d, [e_l^u, .., -e_t, .., -e_n^d], -e_n^u, .., -e_1\}$$

$$arc_1 = \{e_l^u, .., -e_t, .., -e_n^d\}$$

Algorithm 3 - Constructing the arcs

3

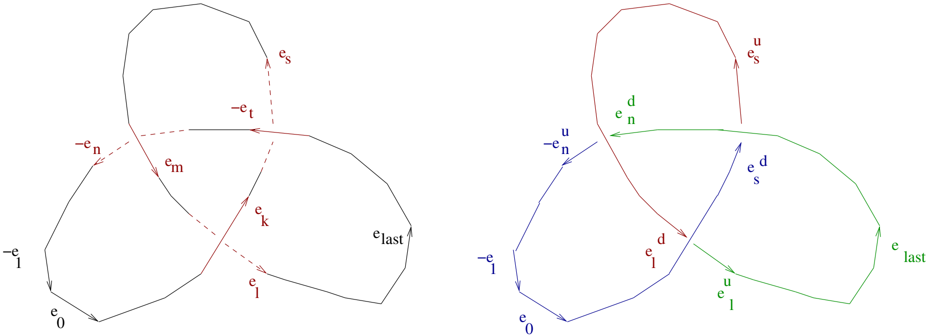


- From the modified loop we compute the arcs until $L_0 = \emptyset$:
  $L_0 = \{e_0, .., e_k, .., e_s^d, -e_n^u, .., -e_1\}$
  $L_0 = \{[e_0, .., e_k, .., e_s^d], [-e_n^u, .., -e_1]\}$
  $arc_2 = \{e_n^u, .., -e_1, e_0, .., e_k, .., e_s^d\}$
- After this step $L_0 = \emptyset$ so the algorithm terminates.

Algorithm 3 - Constructing the arcs

3

- Final output:



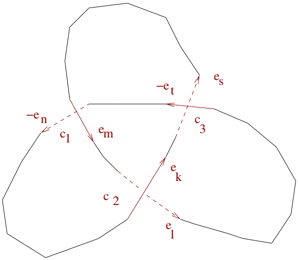
- $E = \{e_0, ..., e_{last}\}$
- $E_I = \{(-e_n, e_m), (e_l, e_k), (e_s, -e_t)\}$ $\Rightarrow$
- $L_0 = \{e_0, ..., e_s, e_l, ..., -e_1\}$

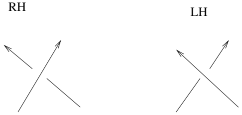
$a_0 = \{e_s^u, ..., e_m, ..., e_l^d\}$

$a_1 = \{e_l^u, .., -e_t, .., -e_n^d\}$

$a_2 = \{e_n^u, .., -e_1, e_0, .., e_k, .., e_s^d\}$

Algorithm 3 - Deciding the type of crossing

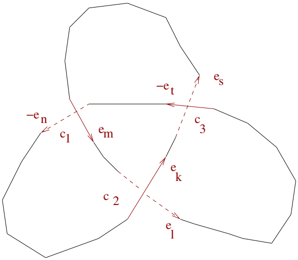


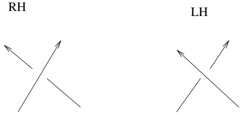


- If $e =$(source,dest)$\in E$ then x.source$<$ x.dest
  If $-e =$(source,dest)$\in E$ then x.source$>$ x.dest
- For any (eUnder,eOver) $\in E_I$ each crossing depends on:
  - the orientation of eUnder, eOver
  - the relation between the slope of eUnder and the slope of eOver
  - there are 2^3 possible cases for deciding the type of crossings.

Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

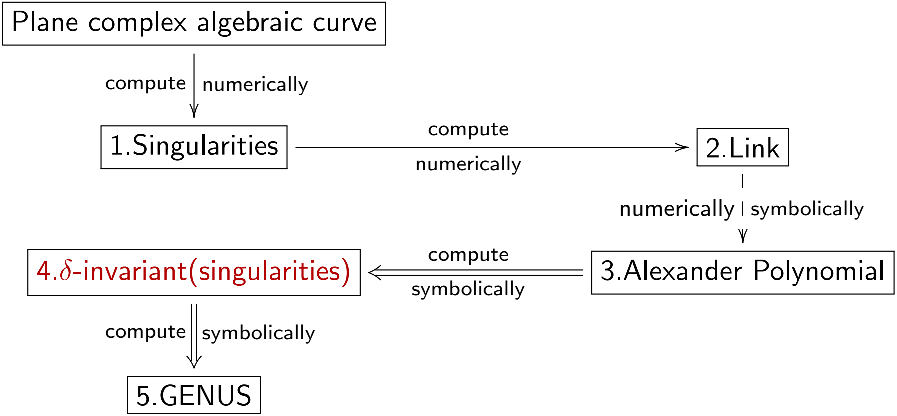
Algorithm 3 - Deciding the type of crossing





- For instance:
- $c_1 = (-e_n, e_m)$ is LH since:
    - x.source($-e_n$)$>x$.dest($-e_n$),
    - x.source(e_m)$<x$.dest(e_m),
    - slope(e_m)$<$slope($-e_n$)
- $c_2 = (e_l, e_k)$ is LH.
- $c_3 = (e_s, -e_t)$ is LH.

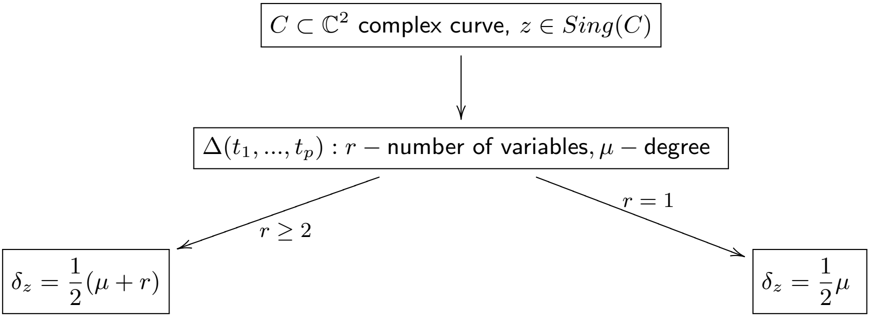
Next



Plane complex algebraic curve
compute
numerically
1.Singularities
compute
numerically
2.Link
numerically
symbolically
3.Alexander Polynomial
compute
symbolically
4.δ-invariant(singularities)
compute
symbolically
5.GENUS

Computing the δ-invariant of the singularity

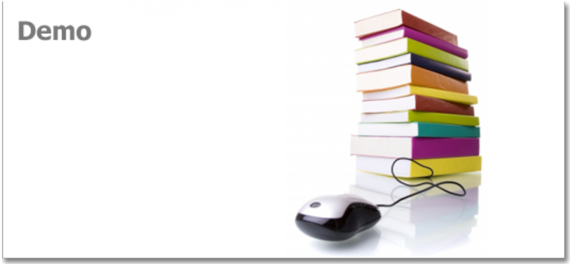
From the Alexander polynomial, we derive the formulae for the δ-invariant: (based on Milnor's research)



Summary

We intend to rename the library *QComplexInvariants* → *GENOM3CK*
Symbolic numeric techniques for *GEN*us c*OM*putation of plane *C*omplex algebrai*C* *C*urves using *K*not theory.

How is the library used?



Summary

Using our library **QComplexInvariants** in Axel we get the results:

Equation	Link	Alex poly, δ-invariant, genus
$z^2 - w^2, \epsilon = 1.0$	Hopf link	$\Delta(t_1) = 1,\ \delta = 1, g = -1$
$z^2 - w^3, \epsilon = 1.0$	Trefoil knot	$\Delta(t_1) = t_1^2 - t_1 + 1, \delta = 1, g = 0$
$z^2 - w^4, \epsilon = 1.0$	2-knots link	$\Delta(t_1, t_2) = t_1 t_2 + 1, \delta = 2, g = -1$
$z^2 - w^5, \epsilon = 1.0$	1-knot link	$\Delta(t_1) = t_1^4 - t_1^3 + t_1^2 - t_1 + 1, \delta = 2, g = 0$
$z^3 - w^3, \epsilon = 1.0$	3-knots link	$\Delta(t_1, t_2, t_3) = -t_1 t_2 t_3 + 1, \delta = 3, g = -2$
$z^4 + z^2 w + w^5, \epsilon = 0.25$	3-knots link	$\Delta(t_1, t_2, t_3) = -t_1^2 t_2^2 t_3 + 1, \delta = 4, g = 2$

Doctoral Program Computational Mathematics Numerical Analysis and Symbolic Computation

Open problem

We perform numeric tests for the original problem with our library

- For small perturbations of the input polynomial we get:

$$\boxed{z^2 - w^3, 1.0001 * z^2 - 1.0001 * w^3} \xrightarrow[\text{solution}]{\text{exact}} \boxed{genus \in \mathbb{Z}}$$

- For tiny perturbations of the input polynomial we get:

$$\boxed{z^2 - w^3 - 0.0001} \xrightarrow[\text{solution}]{\text{approximate}} \boxed{genus \in \mathbb{Z}}$$

Open problem

How to improve the representation to our problem?

Original genus computation problem

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

- Our original genus computation problem is ill-posed since it is infinitely sensitive to perturbation.

Open problem

How to improve the representation to our problem?

We reformulate our problem using **Zeng's** 3 strikes principles:

- the approximate solution is the exact solution of a nearby problem
- the approximate solution is the exact solution of a problem on the nearby pejorative manifold of the highest codimension
- the approximate solution is the exact solution of the nearest problem on the nearby pejorative manifold of the highest codimension

The principle is based on **W. Kahan's** discovery:

- Problems with certain solution structure form a pejorative manifold. The solution is lost when the problem leaves the manifold, but it is preserved when the problem stays on the manifold.

Open problem

How to improve the representation to our problem?

Reformulated genus computation problem

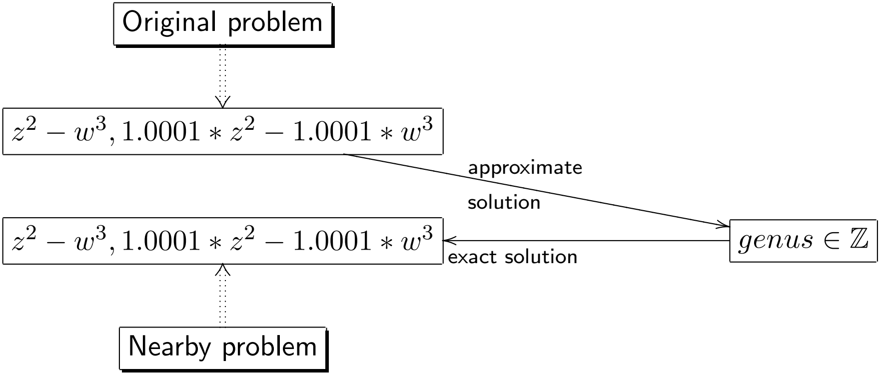
- Input:
  - $\mathbb{C}$ field of complex numbers;
  - $F \in \mathbb{C}[z, w]$ irreducible with coefficients of limited accuracy
  - $C = \{(x, y, u, v) \in \mathbb{R}^4 | F(x + iy, u + iv) = 0\}$ complex curve;
- Output:
  - the approximate genus on a proper pejorative manifold s.t. the computed approximate genus is the exact solution of the nearest polynomial.

- Our symbolic-numerical algorithm solves a "nearby" problem.

Open problem

We perform numeric tests for the reformulated problem with our library

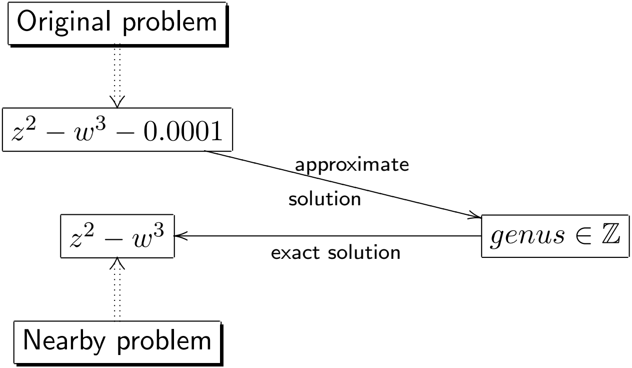
- For small perturbations of the input polynomial we get:



Open problem

We perform numeric tests for the reformulated problem with our library

- For tiny perturbations of the input polynomial we get:



Conclusion

Present work:

- all the steps of the algorithm are now completely automatized;
- together with its main functionality to compute the genus,
- the symbolic-numeric algorithm provides also tools for computation:
  - in knot theory (i.e. diagram of links, Alexander polynomial);
  - in algebraic geometry (i.e. delta invariant, singularities of plane complex algebraic curve);

Conclusion

Future work:

- Analyze the algorithm for numeric input:
  - How to control the error in numerical computation?
  - How to improve the representation to our problem?
- Need to still make more investigations at the frontier between symbolic and numeric computation.

Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

Thank you for your attention.
Questions?