A Symbolic-Numeric Algorithm for Computing the Alexander Polynomial of a Plane Curve Singularity

Mădălina Hodorog[1], Bernard Mourrain[2], Josef Schicho[1]

[1]Johann Radon Institute for Computational and Applied Mathematics,
Doctoral Program "Computational Mathematics"
Johannes Kepler University Linz, Austria
Research Institute for Symbolic Computation

[2]INRIA Sophia-Antipolis, France

12^{th} International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, Timișoara-Romania
September 23, 2010

Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

Table of contents

Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

Motivation

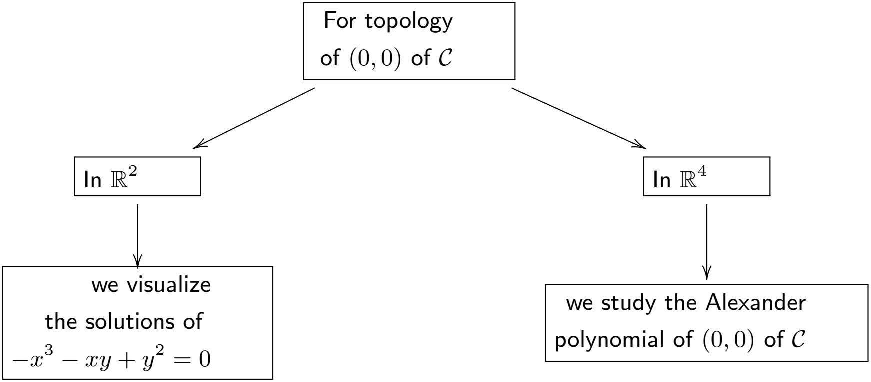
We investigate the topology (i.e. roughly speaking the shape) of plane complex algebraic curves. These curves can be identified with objects in $\mathbb{R}^4$ we cannot visualize! We sketch the equivalent objects in $\mathbb{R}^2$ for a rough "idea"!

Motivation

For instance,
We visualize the topology of the algebraic curve $\mathcal{C} = \{(x, y) | -x^3 - xy + y^2 = 0\}$ in $\mathbb{R}^2$!
We notice an "involved" topology around the point $(0, 0)$, which is called singularity!

Motivation

Roughly, if we consider $\mathcal{C} = \{(x, y) | -x^3 - xy + y^2 = 0\}$ then



Motivation

Our goal is to compute and

Roughly, if we consider $\mathcal{C} = \{(x, y) | -x^3 - xy + y^2 = 0\}$ then

For topology of $(0, 0)$ of $\mathcal{C}$

In $\mathbb{R}^2$

we visualize the solutions of $-x^3 - xy + y^2 = 0$

In $\mathbb{R}^4$

to study the Alexander polynomial of $(0, 0)$ of $\mathcal{C}$

Motivation

Our goal is also to design for:

Rational and floating point numbers → A symbolic-numeric algorithm (in GENOM3CK) (in Axel) → to compute the Alexander polynomial

Because at present (from our knowledge) there exists only for:

Rational numbers → A symbolic algorithm (in Singular) → to compute the Alexander polynomial

Motivation

For instance:

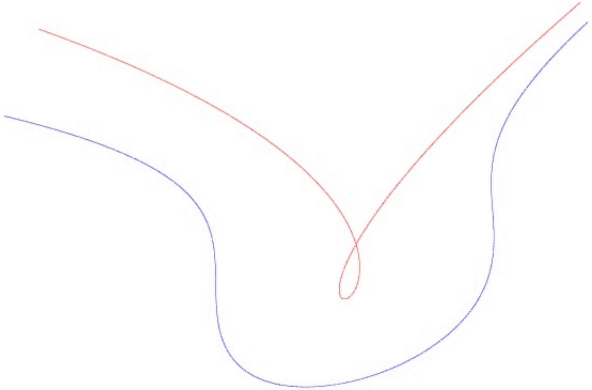
```
                         SINGULAR                             /
 A Computer Algebra System for Polynomial Computations       /   version 3-1-0
                                                           0<
     by: G.-M. Greuel, G. Pfister, H. Schoenemann            \   Mar 2009
FB Mathematik der Universitaet, D-67653 Kaiserslautern        \
> LIB "alexpoly.lib";
// ** loaded /sw/share/Singular/LIB/alexpoly.lib (1.18,2009/04/15)
// ** loaded /sw/share/Singular/LIB/hnoether.lib (1.59,2009/04/15)
// ** loaded /sw/share/Singular/LIB/sing.lib (1.34,2009/04/15)
// ** loaded /sw/share/Singular/LIB/primdec.lib (1.147,2009/04/15)
// ** loaded /sw/share/Singular/LIB/absfact.lib (1.7,2008/07/16)
// ** loaded /sw/share/Singular/LIB/triang.lib (1.14,2009/04/14)
// ** loaded /sw/share/Singular/LIB/matrix.lib (1.48,2009/04/10)
// ** loaded /sw/share/Singular/LIB/nctools.lib (1.54,2009/05/08)
// ** loaded /sw/share/Singular/LIB/ring.lib (1.34,2009/04/15)
// ** loaded /sw/share/Singular/LIB/poly.lib (1.53,2009/04/15)
// ** loaded /sw/share/Singular/LIB/elim.lib (1.34,2009/05/05)
// ** loaded /sw/share/Singular/LIB/general.lib (1.62,2009/04/15)
// ** loaded /sw/share/Singular/LIB/random.lib (1.20,2009/04/15)
// ** loaded /sw/share/Singular/LIB/inout.lib (1.34,2009/04/15)
// ** loaded /sw/share/Singular/LIB/primitiv.lib (1.23,2009/04/15)
> ring r=0,(x,y),ls;
> poly f=-x3-x*y+y2;
> list ALEX=alexanderpolynomial(f);
> def ALEXring=ALEX[1];
> setring ALEXring;
> alexpoly;def precision=10;
1
>
. ring r=(real,precision),(x,y),ls;
// ** redefining r **
> poly f=-x3-x*y+y2-0.01;
> list ALEX=alexanderpolynomial(f);
// ** redefining ALEX **
 Singular cannot factorize over 'real' as ground field
```

Problem specifications

- Input:
  - $f(x,y) \in \mathbb{C}[x,y]$ squarefree with symbolic and numeric coefficients;
  - $\mathcal{C} = \{(x,y) \in \mathbb{C}^2 | f(x,y) = 0\} \subseteq \mathbb{C}^2 \simeq \mathbb{R}^4$ of degree m;
  - $\epsilon \in \mathbb{R}_+^*$ input parameter.
- Output:
  - ϵ-Alexander polynomial of each numerical singularity of $\mathcal{C}$;
- We also compute as output:
  - A set of invariants of $\mathcal{C}$ (numerical singularities, algebraic link, Milnor fibration, Milnor number, δ-invariant of each singularity, genus of $\mathcal{C}$, diagram, crossings, arcs of each algebraic link, etc).

Ill-posedness of the problem

The problem is ill-posed! Small changes in input produce huge changes in the output!
Example. Let $s_1 = (0,0)$ of $\mathcal{C} = \{(x,y) \in \mathbb{R}^2 | -x^3 - xy + y^2 = 0\}$ and $s_2 = (0,0)$ of $\mathcal{D} = \{(x,y) \in \mathbb{R}^2 | -x^3 - xy + y^2 - 0.01 = 0\}$!
The topology of $(0,0)$ is not stable under small changes in input!



The same situation happens in $\mathbb{R}^4$, but we cannot visualize it!

Techniques for dealing with the ill-posedness

How to deal with the **ill-posedness** of **a problem**?

- We construct numerical methods that approximate solutions to ill-posed problems, that are stable under small changes of the input! (i.e. regularization method)
- Similar methods are subjects of *approximate algebraic computation* in order to compute: greatest common divisor of polynomials, root of polynomials, etc.

Techniques for dealing with the ill-posedness

How to deal with the **ill-posedness** of **our problem**?

- *Example.* For $s_1 = (0,0)$ of $\mathcal{C} = \{(x,y) \in \mathbb{R}^4 | -x^3 - xy + y^2 = 0\}$ and $s_2 = (0,0)$ of $\mathcal{D} = \{(x,y) \in \mathbb{R}^4 | -x^3 - xy + y^2 - 0.01 = 0\}$, we compute their ϵ-*algebraic links* denoted $L_\epsilon(s_1), L_\epsilon(s_2)$ (**our research**).

For sufficiently small ϵ, L_ϵ are stable under small changes in the input and they characterize the topology of s_1, s_2 (Milnor's research and **our research**).

From L_ϵ we compute the ϵ-Alexander polynomials (**our research**)
This polynomial is a complete invariant for L_ϵ! (Yamamoto's research)

- If the Alexander polynomials of $L_\epsilon(s_1), L_\epsilon(s_2)$ are equal, then s_1, s_2 have the same topology, else they have different topology!

- Next we compute L_ϵ and ϵ-Alexander polynomial.

Mathematical method and algorithm

Plane complex algebraic curve

↓ compute numerically

Singularities moved in origin

↓ compute numerically-symbolically

Algebraic link and operations on it (ϵ needed)

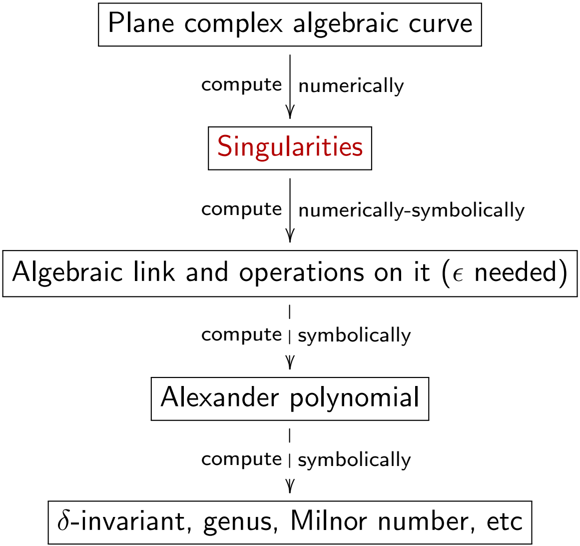
↓ compute symbolically

Alexander polynomial

↓ compute symbolically

δ-invariant, genus, Milnor number, etc

First


Plane complex algebraic curve
compute
numerically
Singularities
compute
numerically-symbolically
Algebraic link and operations on it (ϵ needed)
compute
symbolically
Alexander polynomial
compute
symbolically
δ-invariant, genus, Milnor number, etc

Computing the singularities of the curve

- Input:
  - $f(x,y) \in \mathbb{C}[x,y]$ squarefree with symbolic and numeric coefficients
  - $\mathcal{C} = \{(x,y) \in \mathbb{C}^2 | f(x,y) = 0\}$ complex algebraic curve of degree m.
- Output:
  - $Sing(\mathcal{C}) = \{(x_0,y_0) \in \mathbb{C}^2 | f(x_0,y_0) = 0, \frac{\partial f}{\partial x}(x_0,y_0) = 0, \frac{\partial f}{\partial y}(x_0,y_0) = 0\}$
- Method: We solve the overderminate system in $\mathbb{C}^2$:

$$\begin{cases} f(x_0,y_0) = 0 \\ \frac{\partial f}{\partial x}(x_0,y_0) = 0 \\ \frac{\partial f}{\partial y}(x_0,y_0) = 0 \end{cases} , \qquad (1)$$

using subdivision methods from Axel.

Doctoral Program Computational Mathematics Numerical Analysis and Symbolic Computation

Next

Plane complex algebraic curve

↓ compute numerically

Singularities moved in the origin

↓ compute numerically-symbolically

Algebraic link and operations on it (ϵ needed)

↓ compute symbolically

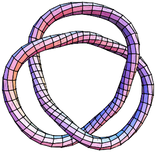
Alexander polynomial

↓ compute symbolically

δ-invariant, genus, Milnor number, etc

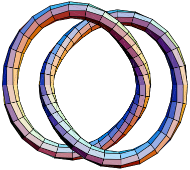
Defining the algebraic link of the singularity

Trefoil Knot



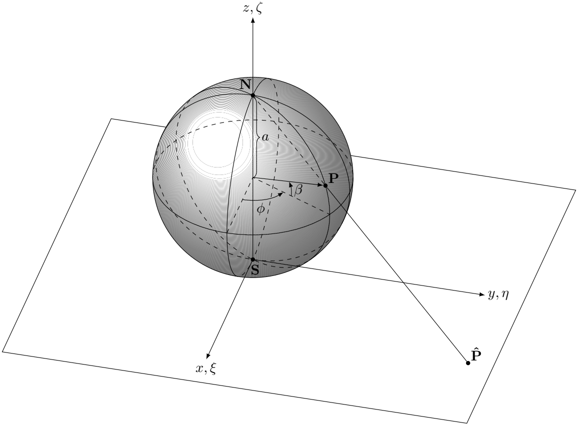
- A **knot** is a piecewise linear or a differentiable simple closed curve in $\mathbb{R}^3$.
- A **link** is a finite union of disjoint knots.
- Links resulted from the intersection of a given curve with the sphere are called **algebraic links**.

Hopf Link



Computing the link of the singularity

- How do we compute the link of a plane curve singularity?
  - use the stereographic projection;



Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

Computing the link of the singularity

1. Let $\mathcal{C} = \{(a,b,c,d) \in \mathbb{R}^4 | f(a,b,c,d) = 0\}$ s.t. $(0,0,0,0) \in Sing(\mathcal{C})$

2. Consider $S_{(0,\epsilon)} := S = \{(a,b,c,d) \in \mathbb{R}^4 | a^2 + b^2 + c^2 + d^2 = \epsilon^2\}$,
   $X = \mathcal{C} \bigcap S \subset \mathbb{R}^4, f(a,b,c,d) = R(a,b,c,d) + iI(a,b,c,d)$

3. For $N \in S \setminus \mathcal{C}$, $\pi : S \setminus \{N\} \to \mathbb{R}^3, (a,b,c,d) \mapsto (u = \frac{a}{\epsilon - d}, v = \frac{b}{\epsilon - d}, w = \frac{c}{\epsilon - d})$,
   $\pi^{-1} : \mathbb{R}^3 \to S \setminus \{N\}$
   $(u,v,w) \mapsto (a,b,c,d) = (\frac{2u\epsilon}{n}, \frac{2v\epsilon}{n}, \frac{2w\epsilon}{n}, \frac{\epsilon(u^2+v^2+w^2-1)}{n})$, with $n = 1+u^2+v^2+w^2$.

4. Denote $\alpha = (\frac{2u\epsilon}{n}, \frac{2v\epsilon}{n}, \frac{2w\epsilon}{n}, \frac{\epsilon(u^2+v^2+w^2-1)}{n})$. Thus $\pi^{-1}(u,v,w) = \alpha$.

5. Compute $\pi(X) = \{(u,v,w) \in \mathbb{R}^3 | \exists (a,b,c,d) = \pi^{-1}(u,v,w,) \in X = \mathcal{C} \cap S\} \Leftrightarrow$
   $\pi(X) = \{(u,v,w) \in \mathbb{R}^3 | f(\alpha) = 0\} = \{(u,v,w) \in \mathbb{R}^3 | R(\alpha) = I(\alpha) = 0\}$ with
   $R, I \in \mathbb{R}[u,v,w]$. $\pi(X)$ is an implicitly defined algebraic curve!
   For small $\epsilon, \pi(X) := L_\epsilon$ is a link (algebraic link) (based on Milnor's research).

Computing the link of the singularity

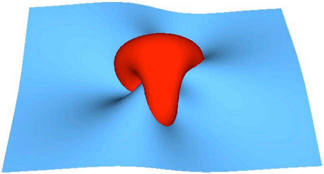
We use Axel for implementation.

- For $\mathcal{C} = \{(x, y) \in \mathbb{C}^2 | x^3 - y^2 = 0\} \subset \mathbb{R}^4, \epsilon = 1$ we compute with the previous method in Axel:

Computing the link of the singularity

We use Axel for implementation.

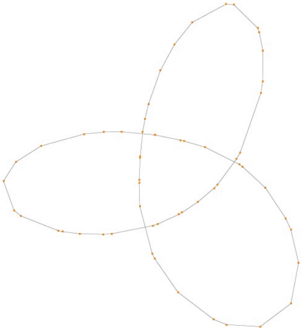
- For $\mathcal{C} = \{(x,y) \in \mathbb{C}^2 | x^3 - y^2 = 0\} \subset \mathbb{R}^4, \epsilon = 1$ we compute with the previous method in Axel:
- $\pi(\mathcal{C} \cap S) = \pi(X) := L_\epsilon =$
  $= \{(u,v,w) \in \mathbb{R}^3 | R(\alpha) = 0, I(\alpha) = 0\}$



Computing the link of the singularity

We use Axel for implementation.

- For $\mathcal{C} = \{(x, y) \in \mathbb{C}^2 | x^3 - y^2 = 0\} \subset \mathbb{R}^4, \epsilon = 1$ we compute with the previous method in Axel:
- $\pi(\mathcal{C} \cap S) = \pi(X) := L_\epsilon =$
  $= \{(u, v, w) \in \mathbb{R}^3 | R(\alpha) = 0, I(\alpha) = 0\}$
- $Graph(L_\epsilon) = \langle \mathcal{V}, \mathcal{E} \rangle$ with
  $\mathcal{V} = \{p = (m, n, q) \in \mathbb{R}^3\}$
  $\mathcal{E} = \{(i, j) | i, j \in \mathcal{V}\}$

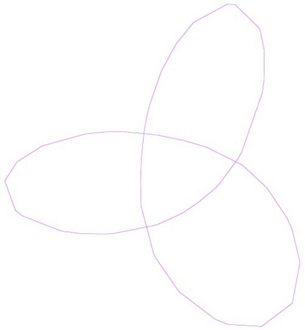


Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

Computing the link of the singularity

We use Axel for implementation.

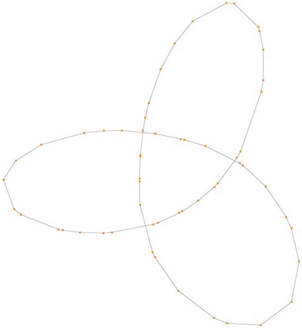
- For $\mathcal{C} = \{(x, y) \in \mathbb{C}^2 | x^3 - y^2 = 0\} \subset \mathbb{R}^4, \epsilon = 1$ we compute with the previous method in Axel:
- $\pi(\mathcal{C} \cap S) = \pi(X) := L_\epsilon =$
  $= \{(u, v, w) \in \mathbb{R}^3 | R(\alpha) = 0, I(\alpha) = 0\}$
- $Graph(L_\epsilon) = \langle \mathcal{V}, \mathcal{E} \rangle$ with
  $\mathcal{V} = \{p = (m, n, q) \in \mathbb{R}^3\}$
  $\mathcal{E} = \{(i, j) | i, j \in \mathcal{V}\}$
- s.t. $Graph(L_\epsilon) \cong_{isotopic} L_\epsilon$



Computing the link of the singularity

We use Axel for implementation.

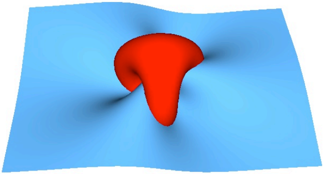
- For $\mathcal{C} = \{(x, y) \in \mathbb{C}^2 | x^3 - y^2 = 0\} \subset \mathbb{R}^4, \epsilon = 1$ we compute with the previous method in Axel:
- $\pi(\mathcal{C} \cap S) = \pi(X) := L_\epsilon =$
  $= \{(u, v, w) \in \mathbb{R}^3 | R(\alpha) = 0, I(\alpha) = 0\}$
- $Graph(L_\epsilon) = \langle \mathcal{V}, \mathcal{E} \rangle$ with
  $\mathcal{V} = \{p = (m, n, q) \in \mathbb{R}^3\}$
  $\mathcal{E} = \{(i, j) | i, j \in \mathcal{V}\}$
- s.t. $Graph(L_\epsilon) \cong_{isotopic} L_\epsilon$
- $Graph(L_\epsilon)$ is a piecewise linear approximation for L_ϵ
- Why Axel? It is the only system to implement a method which returns such an approximation!



Computing the link of the singularity

We use Axel for the implementation. Why Axel?

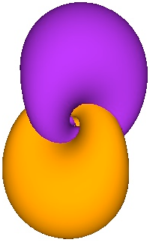
- For $\mathcal{C} = \{(x,y) \in \mathbb{C}^2 | x^3 - y^2 = 0\} \subset \mathbb{R}^4, \epsilon = 1$
- and $L_\epsilon =$
  $= \{(u,v,w) \in \mathbb{R}^3 | R(\alpha) = 0, I(\alpha) = 0\}$



Computing the link of the singularity

We use Axel for the implementation. Why Axel?

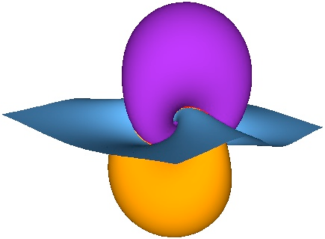
- For $\mathcal{C} = \{(x,y) \in \mathbb{C}^2 | x^3 - y^2 = 0\} \subset \mathbb{R}^4, \epsilon = 1$
- and $L_\epsilon =$
  $= \{(u,v,w) \in \mathbb{R}^3 | R(\alpha) = 0, I(\alpha) = 0\}$
- we also compute (for visualization reasons)
  $\mathcal{S}' = \{(u,v,w) \in \mathbb{R}^3 | R(\alpha) + I(\alpha) = 0\}$
  $\mathcal{S}'' = \{(u,v,w) \in \mathbb{R}^3 | R(\alpha) - I(\alpha) = 0\}$



Computing the link of the singularity

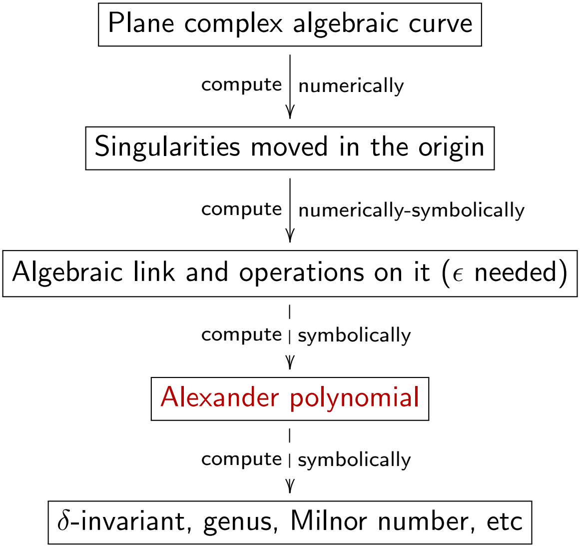
We use Axel for the implementation. Why Axel?

- For $\mathcal{C} = \{(x,y) \in \mathbb{C}^2 | x^3 - y^2 = 0\} \subset \mathbb{R}^4, \epsilon = 1$
- and $L_\epsilon =$
  $= \{(u,v,w) \in \mathbb{R}^3 | R(\alpha) = 0, I(\alpha) = 0\}$
- we also compute (for visualization reasons)
  $\mathcal{S}' = \{(u,v,w) \in \mathbb{R}^3 | R(\alpha) + I(\alpha) = 0\}$
  $\mathcal{S}'' = \{(u,v,w) \in \mathbb{R}^3 | R(\alpha) - I(\alpha) = 0\}$
- L_ϵ is the intersection of any 2 of the surfaces:
  $R(\alpha), I(\alpha)$
  $R(\alpha) + I(\alpha), R(\alpha) - I(\alpha)$



Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

Next



Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

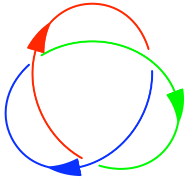
Computing the Alexander polynomial

We give an example to compute the Alexander polynomial Δ_L for a link L with K knots! We need some definitions.

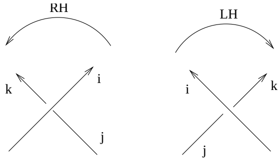
A **diagram** is the image under projection, together with the information on each crossing telling which branch goes over and which under.
An **arc** is the part of a diagram between two undercrossings.

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

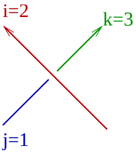


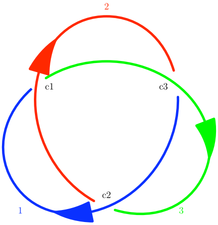
Crossings



Computing the Alexander polynomial of the link

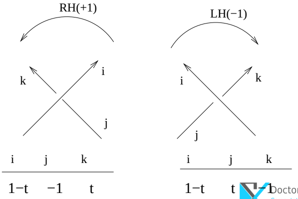
Example: Δ_L for L with $K = 1$ knot (i.e. trefoil knot).





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

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

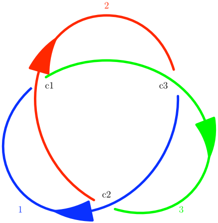


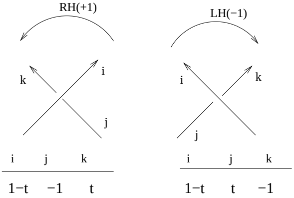
Computing the Alexander polynomial of the link

Example: Δ_L for L with $K = 1$ knot.

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

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



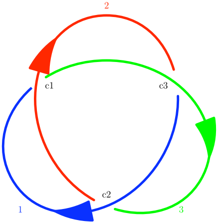


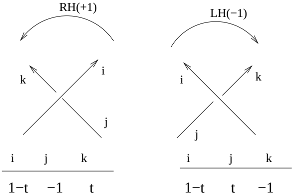
Computing the Alexander polynomial of the link

Example: Δ_L for L with $K = 1$ knot.

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

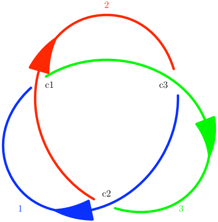
$$P(L) = \begin{pmatrix} 1 & 2 & 3 \\ t & 1-t & -1 \\ & & \end{pmatrix}$$





Computing the Alexander polynomial of the link

Example: Δ_L for L with $K = 1$ knot

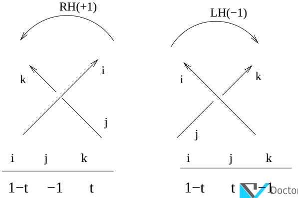


$$P(L) = \begin{pmatrix} t & 1-t & -1 \\ 1-t & -1 & t \\ -1 & t & 1-t \end{pmatrix}$$

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

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

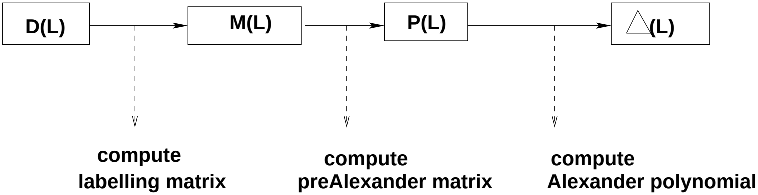
For a link L with $K > 1$ knots and n crossings, $\Delta(L)$ is the gcd of all the $(n-1) \times (n-1)$ minor determinants of $P(L)$.



Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

Computing the Alexander polynomial of the link

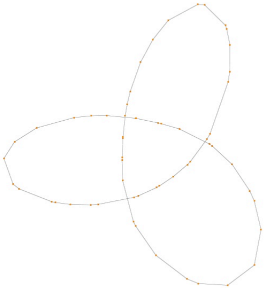
So, the Alexander polynomial is computed in several steps:



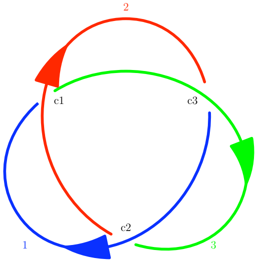
In order to compute it, we need D(L)!

Doctoral Program Computational Mathematics
Numerical Analysis and Symbolic Computation

Computing the Alexander polynomial of the link

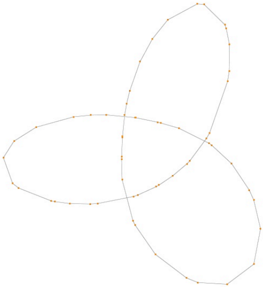
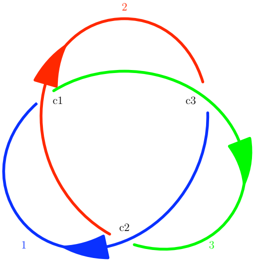
- 

$\overset{?}{\Rightarrow}$



- $G(L_\epsilon) = \langle P, E \rangle$ $D(L_\epsilon)$
- We need to transform the graph data structure $G(L_\epsilon)$ returned by Axel into the diagram of the algebraic link $D(L_\epsilon)$.

Computing the Alexander polynomial of the link

-  $\overset{?}{\Rightarrow}$ 

- We developed several computational geometry and combinatorial algorithms for this! (M. Hodorog, J.Schicho. *Computational geometry and combinatorial algorithms for the genus computation problem.* DK 10-07 Report).

Implementation

- *Axel* free algebraic geometric modeler (INRIA Sophia-Antipolis) [a]

http://axel.inria.fr/

[a] Acknowledgements: Julien Wintz

Implementation

- *Axel* free algebraic geometric modeler (INRIA Sophia-Antipolis) [a]
  - written in *C*++, Qt Script for Applications (QSA);

[a] Acknowledgements: Julien Wintz

Implementation

- *Axel* free algebraic geometric modeler (INRIA Sophia-Antipolis) [a]
  - written in *C++*, Qt Script for Applications (QSA);
- GENOM3CK-our library in Axel (GEnus cOMputation of plane Complex algebraiC Curves with Knot theory). Support: `http://people.ricam.oeaw.ac.at/m.hodorog/software.html` and `madalina.hodorog@oeaw.ac.at`

[a] Acknowledgements: Julien Wintz

Implementation

- *Axel* free algebraic geometric modeler (INRIA Sophia-Antipolis) [a]
  - written in *C*++, Qt Script for Applications (QSA);
- GENOM3CK-our library in Axel (GEnus cOMputation of plane Complex algebraiC Curves with Knot theory). Support: `http://people.ricam.oeaw.ac.at/m.hodorog/software.html` and `madalina.hodorog@oeaw.ac.at`
- Version 0.2 of GENOM3CK is released!

[a] Acknowledgements: Julien Wintz

Analysis of the algorithm

With the notations:

- E the symbolic algorithm to compute the Alexander polynomial, which is ill-posed.
- A_ϵ the symbolic-numeric algorithm to compute the ϵ-Alexander polynomial Δ_ϵ from the ϵ-algebraic link L_ϵ.
  - For input polynomial f, $A_\epsilon(f)$ returns as output Δ_ϵ.
  - For perturbed f_δ (for any $\delta : ||f - f_\delta|| < \delta$), $A_\epsilon(f_\delta)$ returns as output Δ_ϵ^δ.

and based on:

- Milnor's theorem, i.e. if $\epsilon \to 0$ then Δ_ϵ tends to the exact solution (**convergence for exact data**)
- and on general results from regularization theory (adapted to our case).

The algorithm A_ϵ is a regularization, i.e.:

Analysis of the algorithm

With the notations:

- E the symbolic algorithm to compute the Alexander polynomial, which is ill-posed.
- A_ϵ the symbolic-numeric algorithm to compute the ϵ-Alexander polynomial Δ_ϵ from the ϵ-algebraic link L_ϵ.
  - For input polynomial f, $A_\epsilon(f)$ returns as output Δ_ϵ.
  - For perturbed f_δ (for any $\delta : ||f - f_\delta|| < \delta$), $A_\epsilon(f_\delta)$ returns as output Δ_ϵ^δ.

and based on:

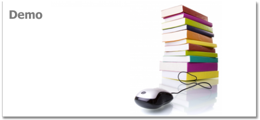
- Milnor's theorem, i.e. if $\epsilon \rightarrow 0$ then Δ_ϵ tends to the exact solution (**convergence for exact data**)
- and on general results from regularization theory (adapted to our case).

The algorithm A_ϵ is a regularization, i.e.:

- Δ_ϵ^δ depends continuously on the perturbed input polynomial f_δ (**continuity**);
- If $\delta \rightarrow 0$ and ϵ is chosen appropiately, then Δ_ϵ^δ tends to the exact solution (**convergence for perturbed data**).

Demo (Numeric and Symbolic Examples)

Equation in $\mathbb{R}^4$	Box
$-x^3 - xy + y^2, \epsilon = 1.00$	$[-4, 4, -6, 6, -6, 6]$
$-x^3 - xy + y^2, \epsilon = 0.25$	$[-4, 4, -6, 6, -6, 6]$
$-x^3 - xy + y^2 - 0.01, \epsilon = 1.00$	$[-4, 4, -6, 6, -6, 6]$
$-x^3 - xy + y^2 - 0.01, \epsilon = 0.25$	$[-4, 4, -6, 6, -6, 6]$



Conclusion and future work

✓ DONE:

- automatization of symbolic-numeric algorithms for plane curves in GENOM3CK (i.e. algorithm to compute the Alexander polynomial);
- describe algorithms with principles from regularization theory;

✗ TO DO's:

Conclusion and future work

✓ DONE:

- automatization of symbolic-numeric algorithms for plane curves in GENOM3CK (i.e. algorithm to compute the Alexander polynomial);
- describe algorithms with principles from regularization theory;
- test experiments show that the algorithm has the continuity and the convergence for perturbed data properties;
- proofs for continuity and convergence for perturbed data properties are constructed.

✗ TO DO's:

Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

Conclusion and future work

✓ DONE:

- automatization of symbolic-numeric algorithms for plane curves in GENOM3CK (i.e. algorithm to compute the Alexander polynomial);
- describe algorithms with principles from regularization theory;
- test experiments show that the algorithm has the continuity and the convergence for perturbed data properties;
- proofs for continuity and convergence for perturbed data properties are constructed.

✗ TO DO's:

- finalize the proof for convergence for perturbed data property of the algorithm;

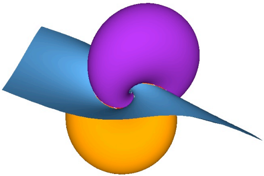
Conclusion and future work

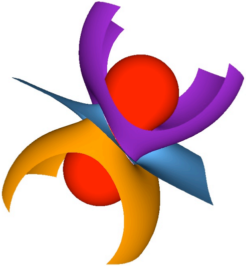
✓ DONE:

- automatization of symbolic-numeric algorithms for plane curves in GENOM3CK (i.e. algorithm to compute the Alexander polynomial);
- describe algorithms with principles from regularization theory;
- test experiments show that the algorithm has the continuity and the convergence for perturbed data properties;
- proofs for continuity and convergence for perturbed data properties are constructed.

✗ TO DO's:

- finalize the proof for convergence for perturbed data property of the algorithm;
- include other operations, i.e. from knot theory, algebraic geometry.





Thank you for your attention.
Questions?