The genus computation problem: symbolic-numeric solutions and beyond

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,
Doctoral Program "Computational Mathematics"
Johannes Kepler University Linz, Austria

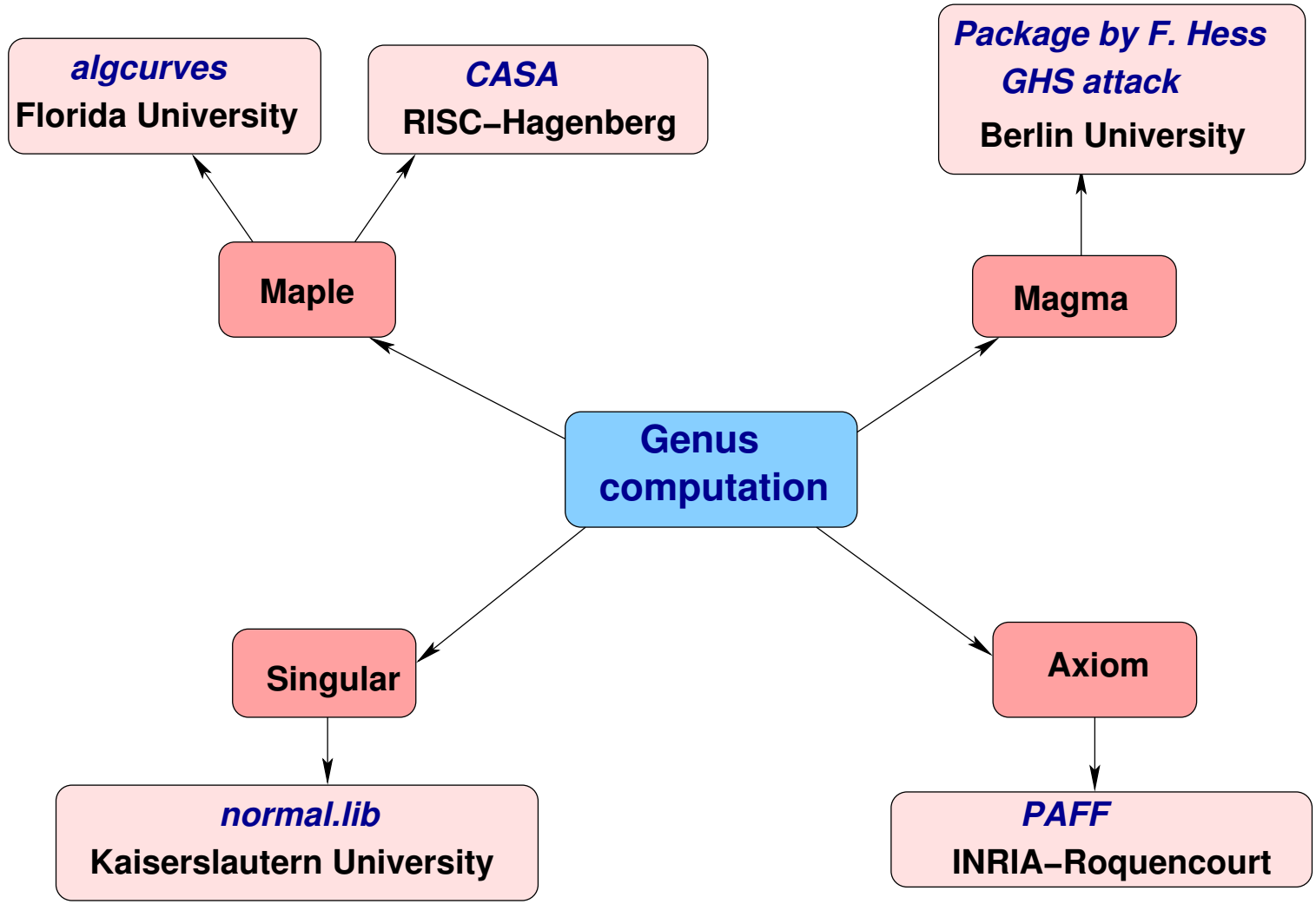
[2] INRIA Sophia-Antipolis, France

SAGA Winter School Auron, France
March 17, 2010

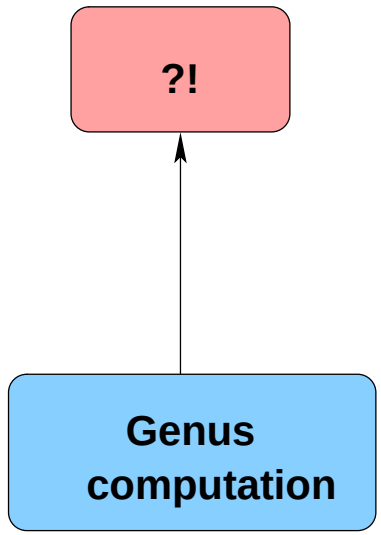
Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

Table of contents

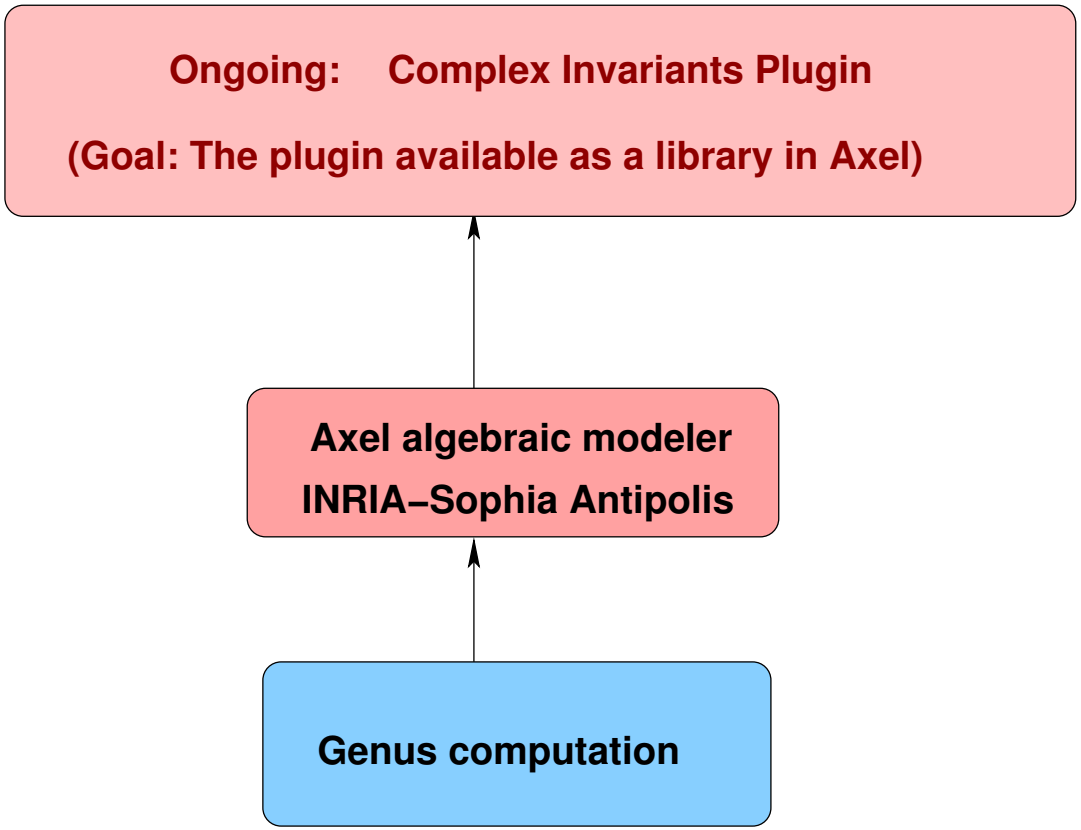
Symbolic Algorithms:



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 (in the group of R. Sendra).

What?

- Input:
  - $F \in \mathbb{C}[x, y]$ squarefree with coefficients of limited accuracy:
  - $C = \{(x, y) \in \mathbb{C}^2 | F(x, y) = 0\}$ complex algebraic curve of degree m.
  - $\epsilon \in \mathbb{R}_+^*$ a non-zero positive real number, the input parameter.
- Output:
  - "approximate" $genus(C)$, i.e.
    the lowest possible genus of a curve defined by a "nearby" polynomial, s.t.

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

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

What?

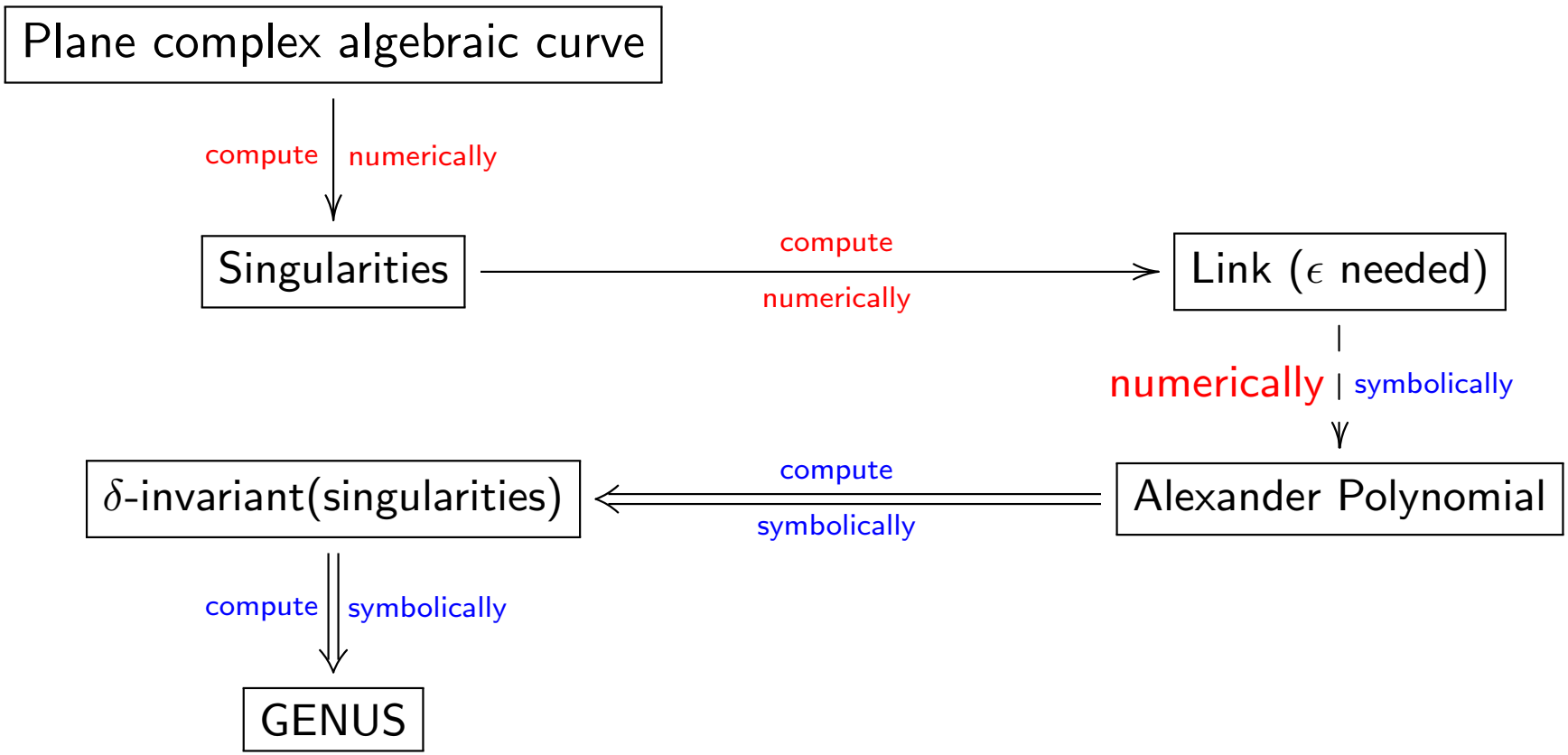
- Input:
  - $F \in \mathbb{C}[x, y]$ squarefree with coefficients of limited accuracy:
    - either exact data, i.e. integers or rational numbers: $1, -2, \frac{1}{2}$.
    - or inexact data, i.e. real numbers. For 1.001 we associate a tolerance of 10^{-3}, i.e. the last digit is uncertain.
  - $C = \{(x, y) \in \mathbb{C}^2 | F(x, y) = 0\}$ complex algebraic curve of degree m.
  - $\epsilon \in \mathbb{R}_+^*$ a non-zero positive real number, the input parameter.
- Output:
  - "approximate" $genus(C)$, i.e.
    the lowest possible genus of a curve defined by a "nearby" polynomial, s.t.

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

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

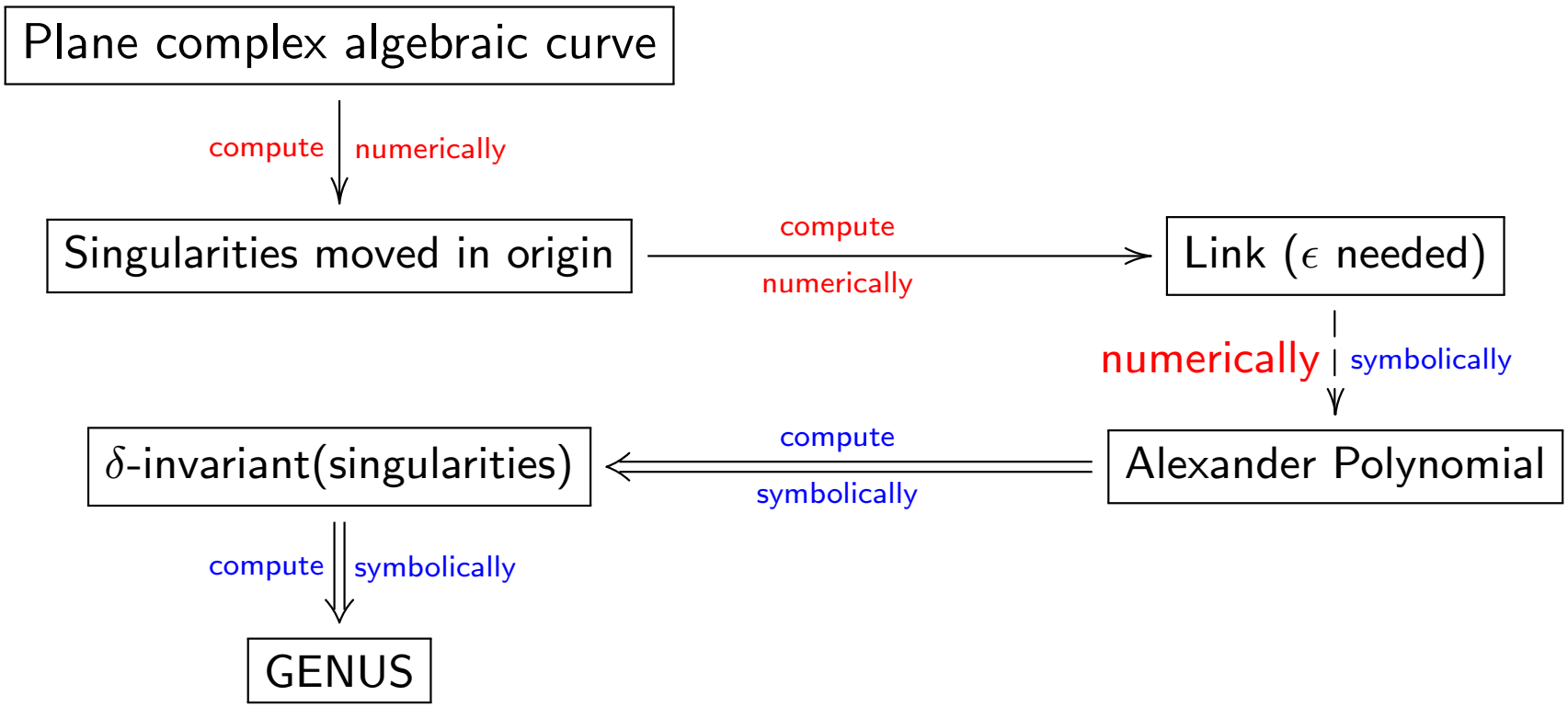
How?

- Strategy for computing the genus



How?

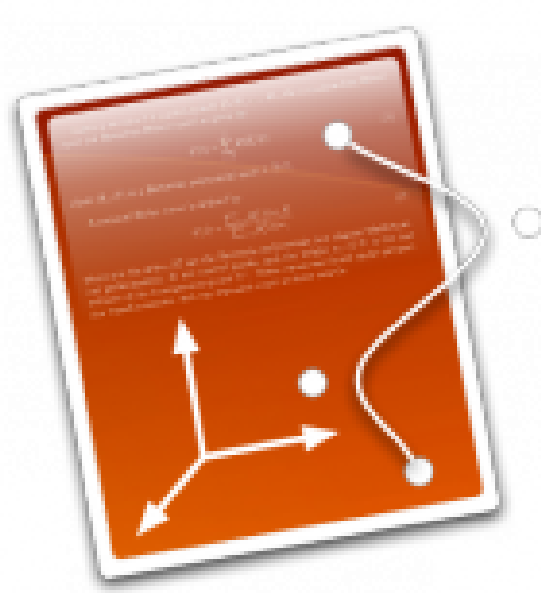
- Strategy for computing the genus



Solving the problem

Implementation of the algorithm

- *Axel* algebraic geometric modeler [a]



[a] Acknowledgements: Julien Wintz

Solving the problem

Implementation of the algorithm

- *Axel* algebraic geometric modeler [a]
  - developed by *Galaad* team (INRIA Sophia-Antipolis);

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

[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);

[a] Acknowledgements: Julien Wintz

Solving the problem

Implementation of the algorithm

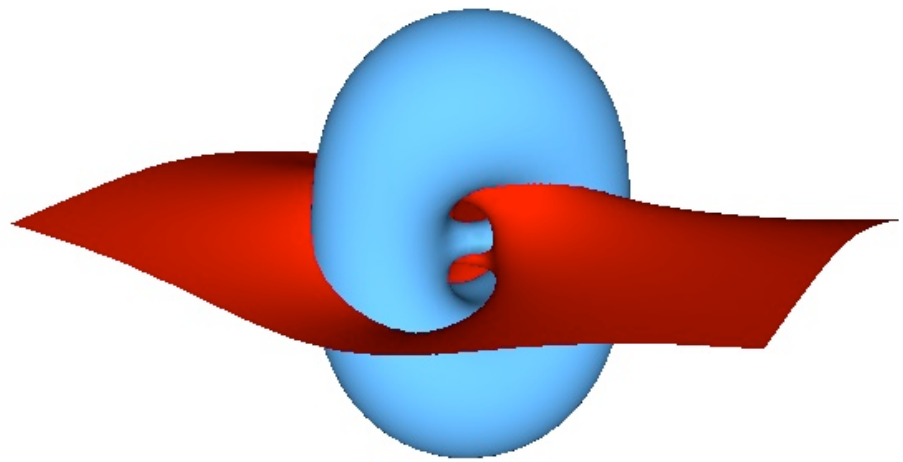
- *Axel* algebraic geometric modeler [a]
  - developed by *Galaad* team (INRIA Sophia-Antipolis);
  - written in *C++*, Qt Script for Applications (QSA);
  - uses *mmx* libraries (shape, realroot);

[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);
  - uses *mmx* libraries (shape, realroot);
  - provides algebraic tools for:
    - implicit surfaces;

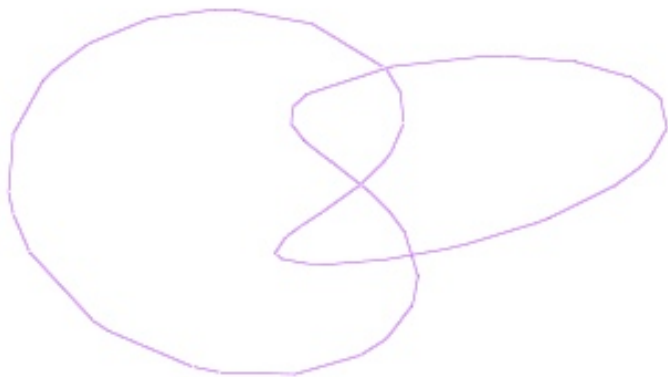


[a] Acknowledgements: Julien Wintz

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);
  - uses *mmx* libraries (shape, realroot);
  - provides algebraic tools for:
    - implicit surfaces;
    - implicit curves.

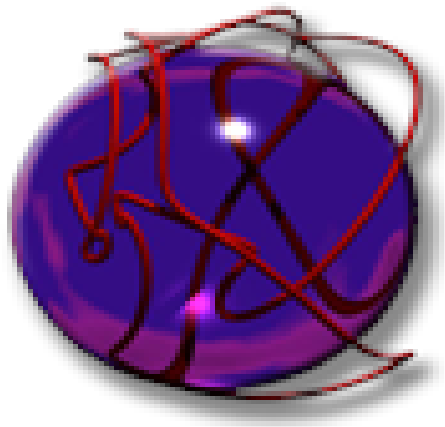


[a] Acknowledgements: Julien Wintz

Solving the problem

Implementation of the algorithm

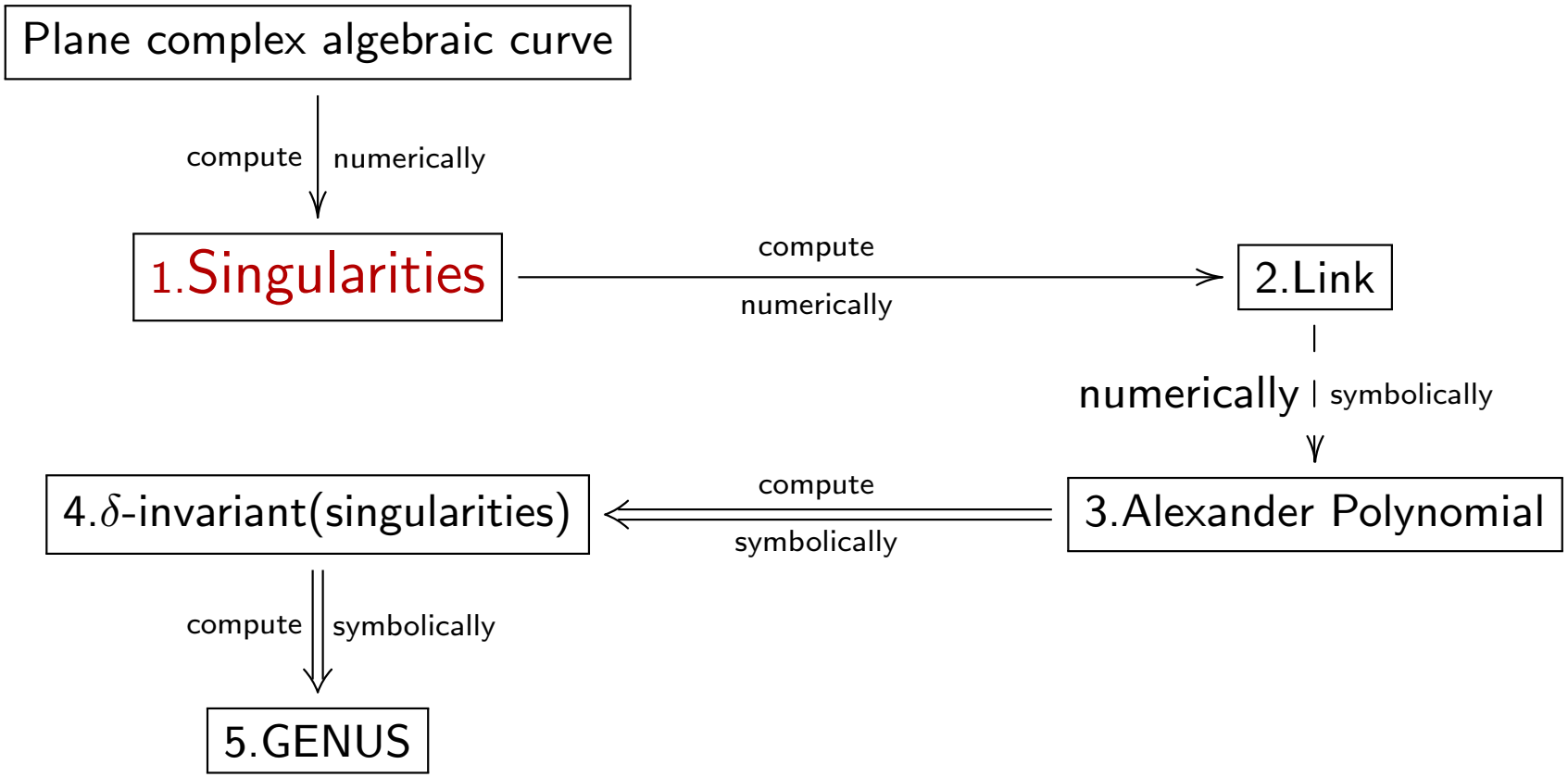
- *Axel* algebraic geometric modeler [a]
  - developed by *Galaad* team (INRIA Sophia-Antipolis);
  - written in *C++*, Qt Script for Applications (QSA);
  - uses *mmx* libraries (shape, realroot);
  - provides algebraic tools for:
    - implicit surfaces;
    - implicit curves.
  - free, available at:



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:
  - $F \in \mathbb{C}[x, y]$
  - $C = \{(x, y) \in \mathbb{C}^2 | F(x, y) = 0\}$
- Output:
  - $Sing(C) = \{(x_0, y_0) \in \mathbb{C}^2 | F(x_0, y_0) = 0, \frac{\partial F}{\partial x}(x_0, y_0) = 0, \frac{\partial F}{\partial y}(x_0, y_0) = 0\}$

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

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

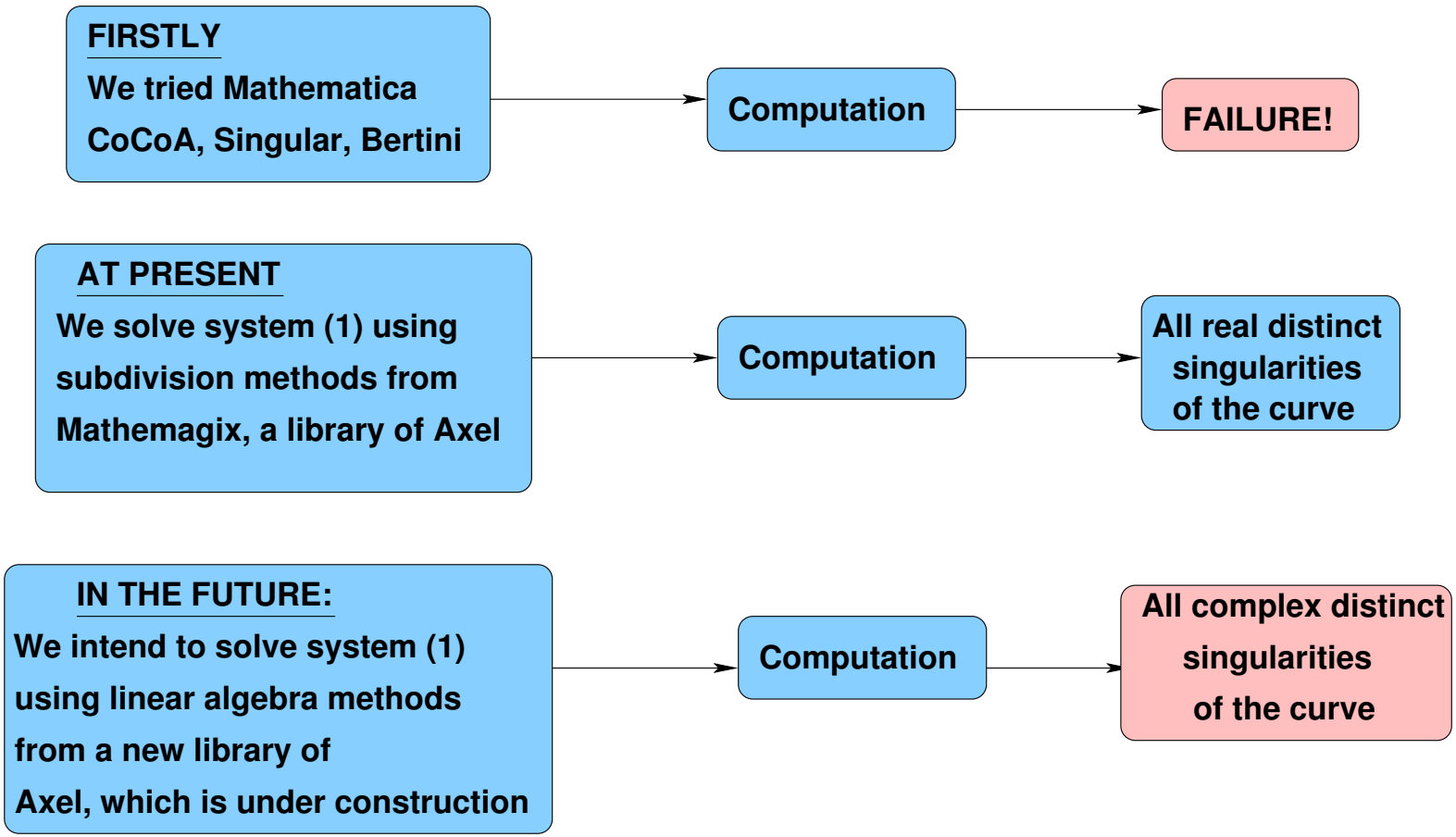
Computing the singularities of the curve

or in $\mathbb{R}^4$: $F(x,y) = F(a+ib, c+id) = s(a,b,c,d) + it(a,b,c,d)$

$$\begin{cases} s(a_0,b_0,c_0,d_0) = 0 \\ t(a_0,b_0,c_0,d_0) = 0 \\ \\ \dfrac{\partial s}{\partial a}(a_0,b_0,c_0,d_0) = 0 \\ \\ \dfrac{\partial t}{\partial a}(a_0,b_0,c_0,d_0) = 0 \\ \\ \dfrac{\partial s}{\partial c}(a_0,b_0,c_0,d_0) = 0 \\ \\ \dfrac{\delta t}{\delta c}(a_0,b_0,c_0,d_0) = 0 \end{cases}, \tag{2}$$

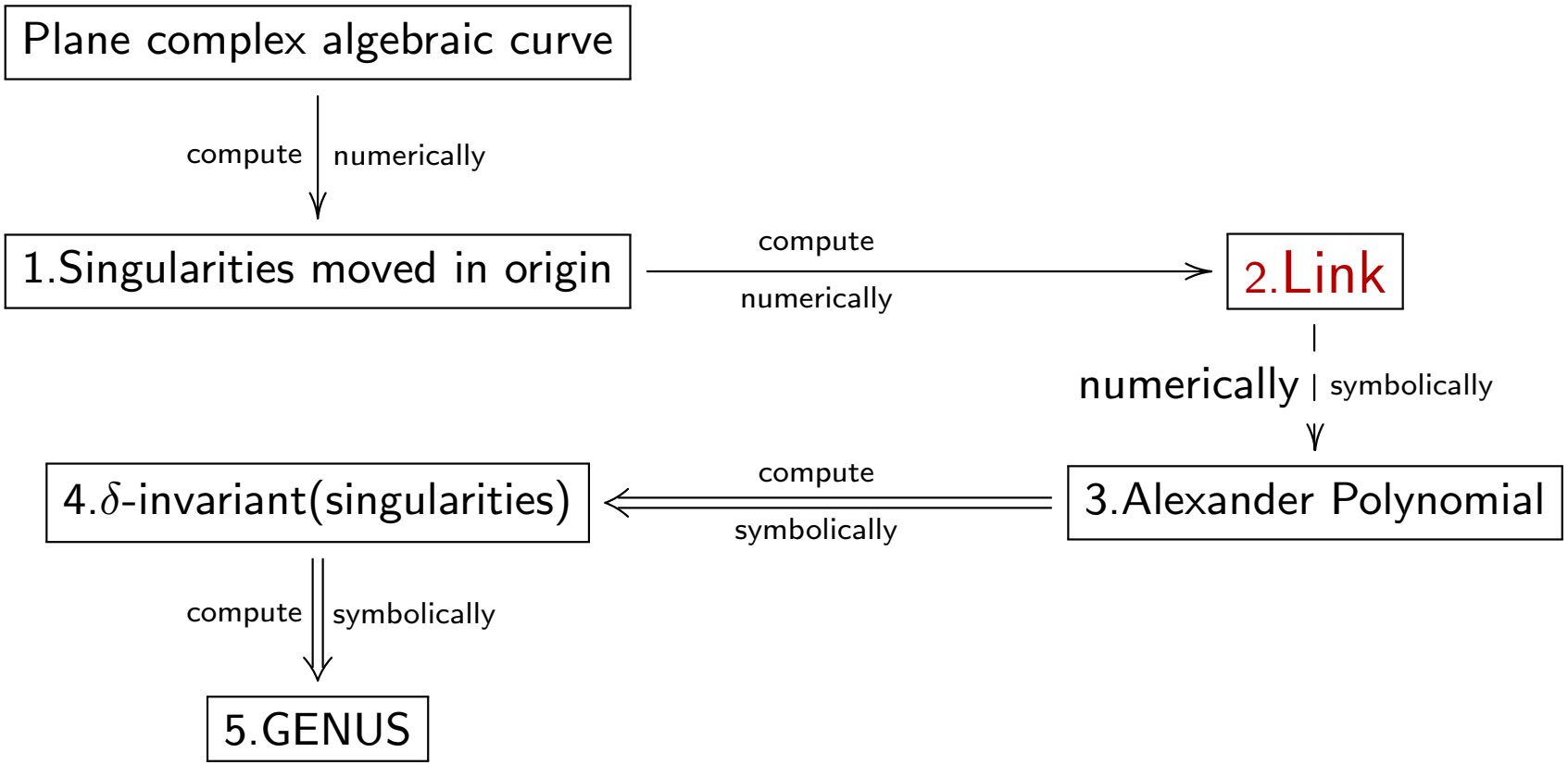
Computing the singularities of the curve

For input polynomials with coefficients of limited accuracy



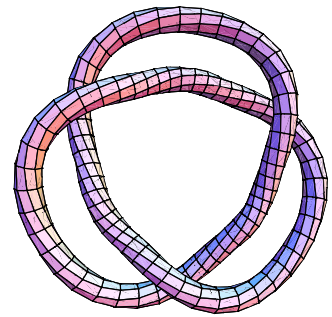
Note: so far this is an open problem.

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

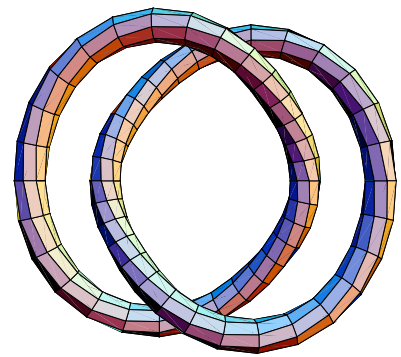
Knot theory - preliminaries

Trefoil Knot



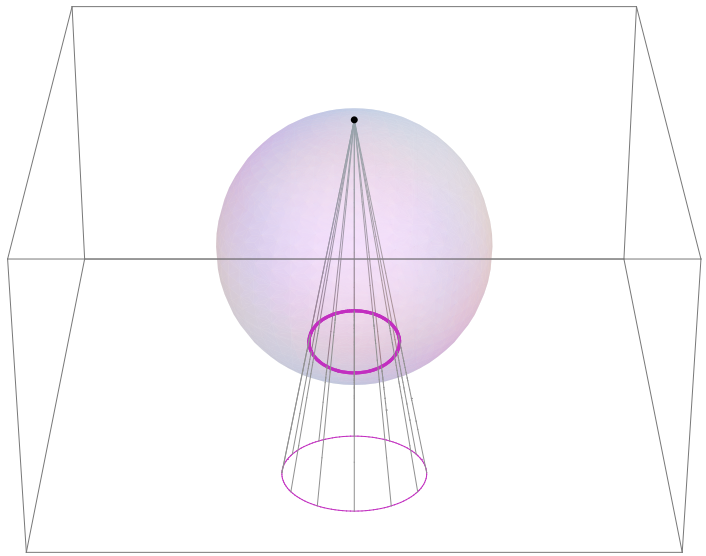
- A **knot** is a simple closed curve in $\mathbb{R}^3$.
- A **link** is a finite union of disjoint knots.
- Links resulted from the intersection of a given curve with the sphere are called **algebraic links**.

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

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

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

4. Compute $f(X) = \{(u,v,w) \in \mathbb{R}^3 | F(\frac{2u\epsilon}{n}, \frac{2v\epsilon}{n}, ...) = 0\} \Leftrightarrow$
$f(X) = \{(u,v,w) \in \mathbb{R}^3 | ReF(...) = 0, ImF(...) = 0\}$
For small ϵ, $f(X)$ is a link.

Computing the link of the singularity

Why Axel?

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

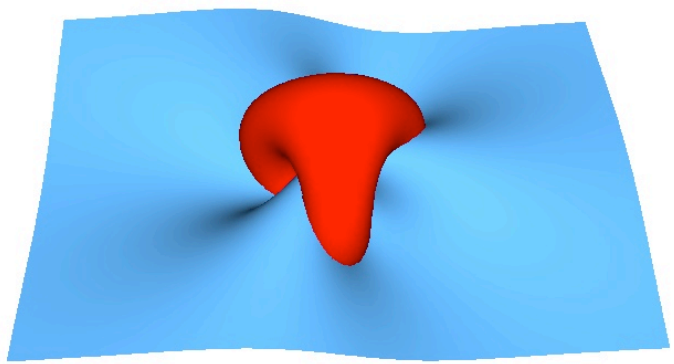
- For $C^4 = \{(x, y) \in \mathbb{C}^2 | x^3 - y^2 = 0\} \subset \mathbb{R}^4$ get

Computing the link of the singularity

Why Axel?

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

- For $C^4 = \{(x, y) \in \mathbb{C}^2 | x^3 - y^2 = 0\} \subset \mathbb{R}^4$ get
- $f(C^4 \cap S) := C =$
  $= \{(u, v, w) \in \mathbb{R}^3 | ReF(...) = 0, ImF(...) = 0\}$

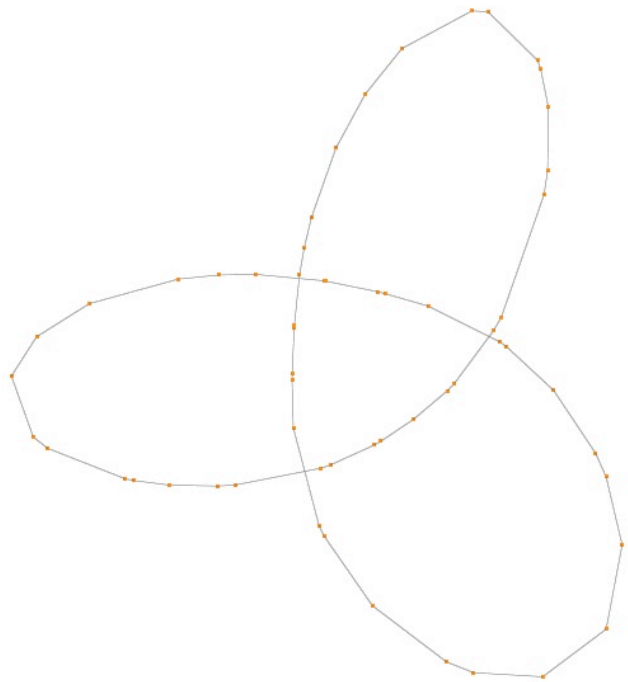


Computing the link of the singularity

Why Axel?

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

- For $C^4 = \{(x, y) \in \mathbb{C}^2 | x^3 - y^2 = 0\} \subset \mathbb{R}^4$ get
- $f(C^4 \cap S) := C =$
  $= \{(u, v, w) \in \mathbb{R}^3 | ReF(...) = 0, ImF(...) = 0\}$
- compute $Graph(C) = \langle \mathcal{V}, \mathcal{E} \rangle$ with
  $\mathcal{V} = \{p = (m, n, q) \in \mathbb{R}^3\}$
  $\mathcal{E} = \{(i, j) | i, j \in \mathcal{V}\}$

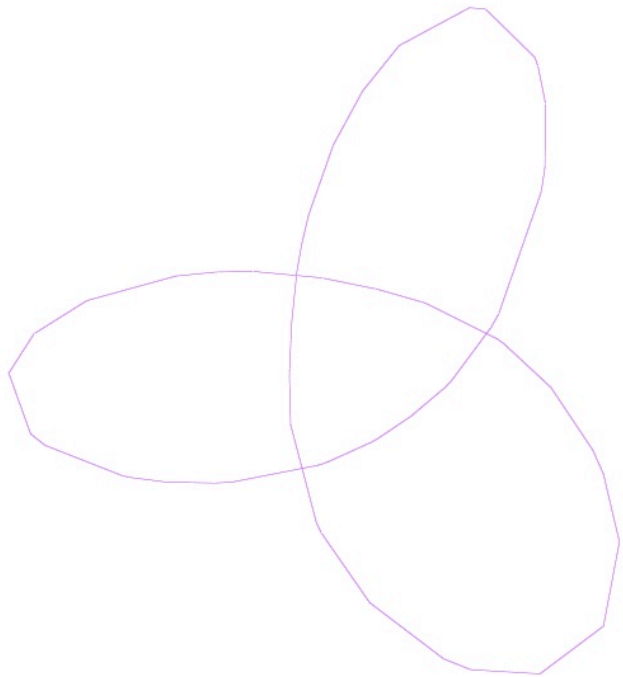


Computing the link of the singularity

Why Axel?

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

- For $C^4 = \{(x, y) \in \mathbb{C}^2 | x^3 - y^2 = 0\} \subset \mathbb{R}^4$ get
- $f(C^4 \cap S) := C =$
  $= \{(u, v, w) \in \mathbb{R}^3 | ReF(...) = 0, ImF(...) = 0\}$
- 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. $Graph(C) \cong_{isotopic} C$

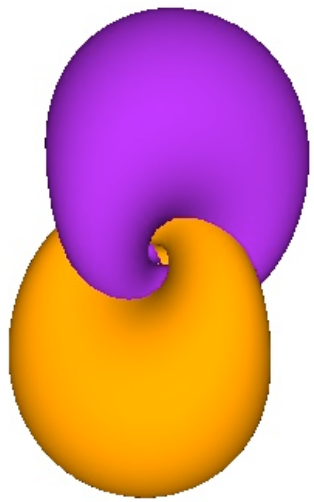


Computing the link of the singularity

Why Axel?

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

- For $C^4 = \{(x, y) \in \mathbb{C}^2 | x^3 - y^2 = 0\} \subset \mathbb{R}^4$ get
- $f(C^4 \cap S) := C =$
  $= \{(u, v, w) \in \mathbb{R}^3 | ReF(...) = 0, ImF(...) = 0\}$
- 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. $Graph(C) \cong_{isotopic} C$
- also compute (for visualization reasons only)
  $S^{'} = \{(u, v, w) \in \mathbb{R}^3 | ReF(...) + ImF(...) = 0\}$
  $S^{''} = \{(u, v, w) \in \mathbb{R}^3 | Re(F) - ImF(...) = 0\}$

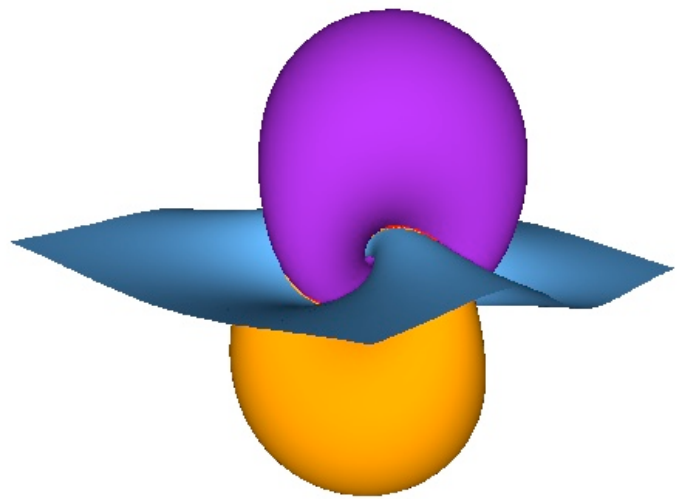


Computing the link of the singularity

Why Axel?

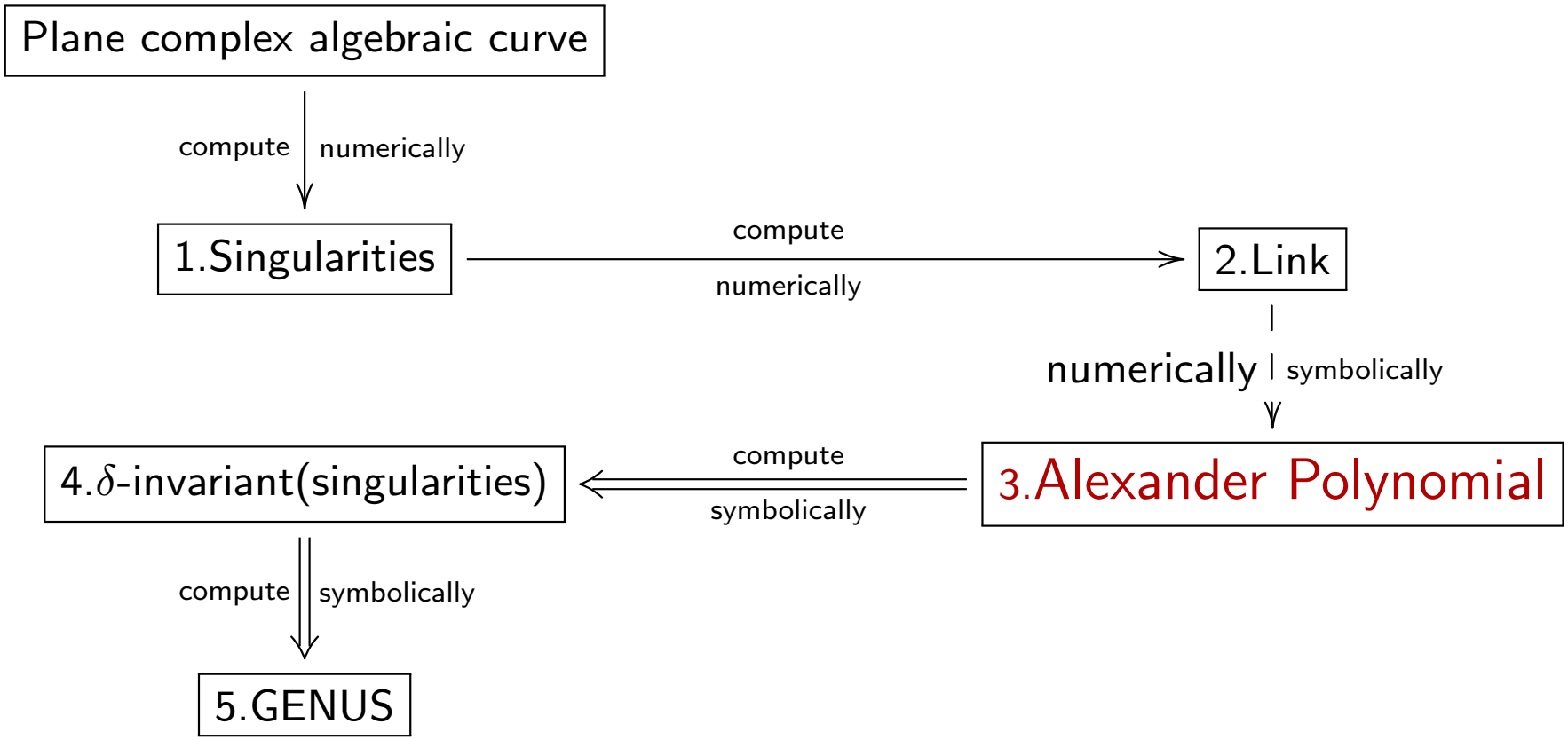
It computes numerically the certified topology of smooth implicit curves in $\mathbb{R}^3$

- For $C^4 = \{(x, y) \in \mathbb{C}^2 | x^3 - y^2 = 0\} \subset \mathbb{R}^4$ get
- $f(C^4 \cap S) := C =$
  $= \{(u, v, w) \in \mathbb{R}^3 | ReF(...) = 0, ImF(...) = 0\}$
- 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. $Graph(C) \cong_{isotopic} C$
- also compute (for visualization reasons only)
  $S^{'} = \{(u, v, w) \in \mathbb{R}^3 | ReF(...) + ImF(...) = 0\}$
  $S^{''} = \{(u, v, w) \in \mathbb{R}^3 | Re(F) - ImF(...) = 0\}$
- C is the intersection of any 2 of the surfaces:
  $ReF(...), ImF(...)$
  $ReF(...) + ImF(...), ReF(...) - ImF(...)$



Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

Next

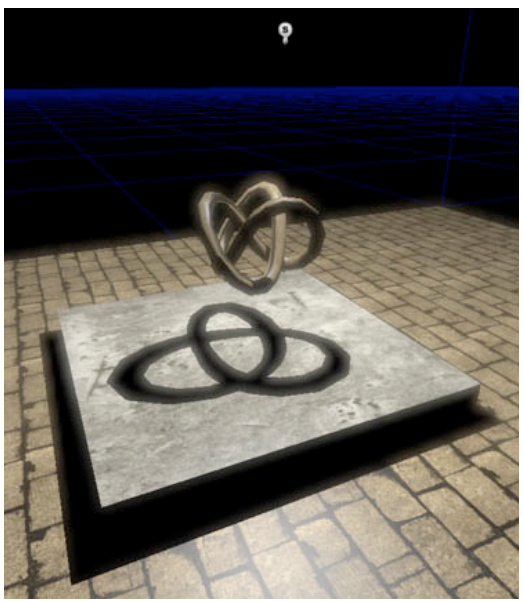

Plane complex algebraic curve
compute
numerically
1.Singularities
compute
numerically
2.Link
numerically
symbolically
4.δ-invariant(singularities)
compute
symbolically
3.Alexander Polynomial
compute
symbolically
5.GENUS

Knot theory - preliminaries

The Alexander polynomial (1928) depends on the fundamental group of the complement of the knot in $\mathbb{R}^3$.

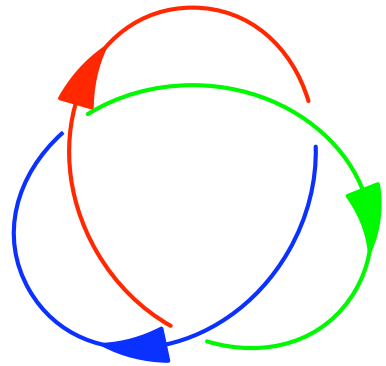
Definition. Let L be a link with n components. The multivariate Alexander polynomial is a Laurent polynomial $\Delta_L \in \mathbb{Z}[t_0, ..., t_n, t_0^{-1}, ..., t_n^{-1}]$, which is defined up to a factor of $\pm t_0^{k_0} ... t_n^{k_n}, k_i \in \mathbb{Z}, \forall i \in \{0, ..., n\}$.

Note. The Alexander polynomial is a complete invariant for the algebraic links (Yamamoto 1984).



Knot theory - preliminaries

Diagram and arcs

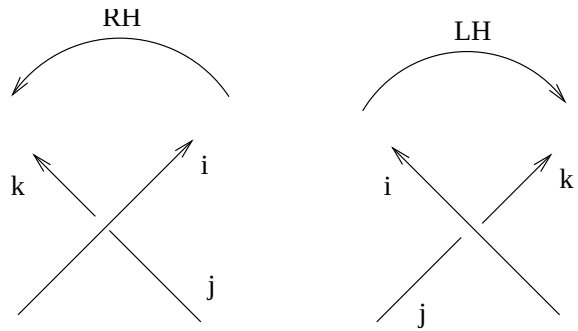


A knot projection is a **regular projection** if no three points on the knot project to the same point, and no vertex projects to the same point as any other point on the knot.

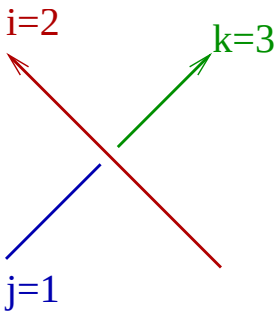
A **diagram** is the image under regular projection, together with the information on each crossing telling which branch goes over and which under.
An **arc** is the part of a diagram between two undercrossings.

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

Crossings

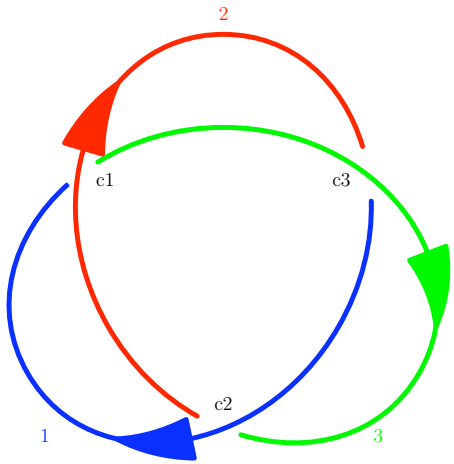


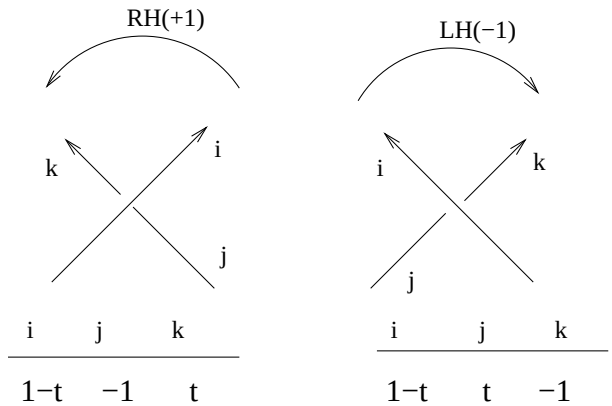
Computing the Alexander polynomial of the link



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

$$P(L) = \left(\begin{array}{c} \end{array} \right)$$

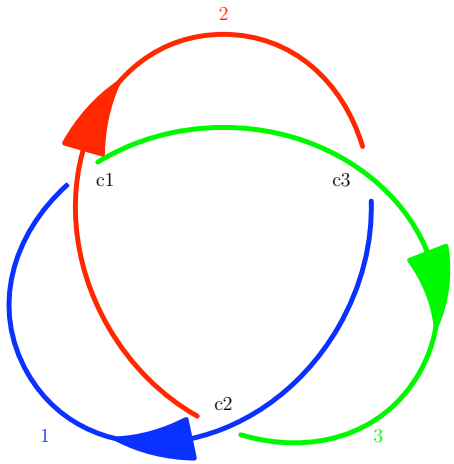


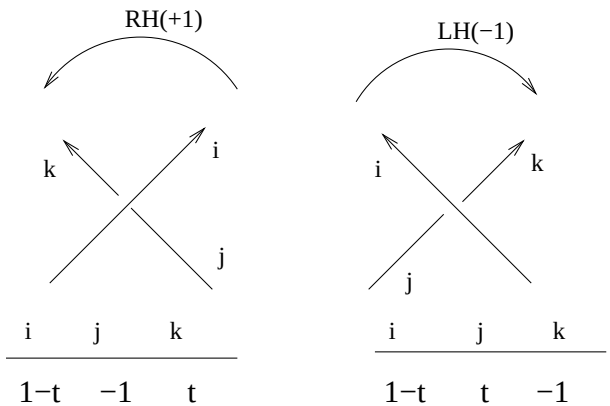


Computing the Alexander polynomial of the link

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

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

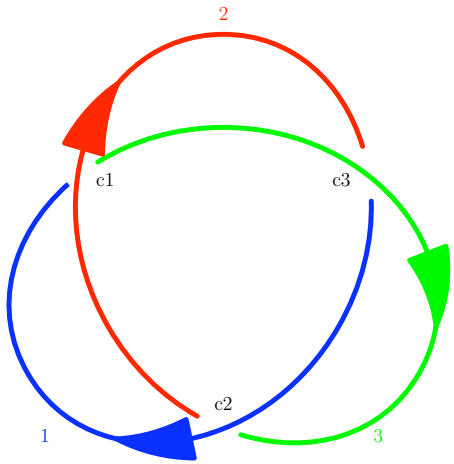


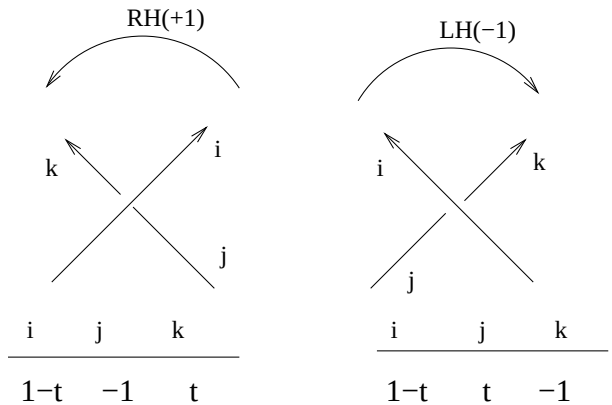


Computing the Alexander polynomial of the link

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

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





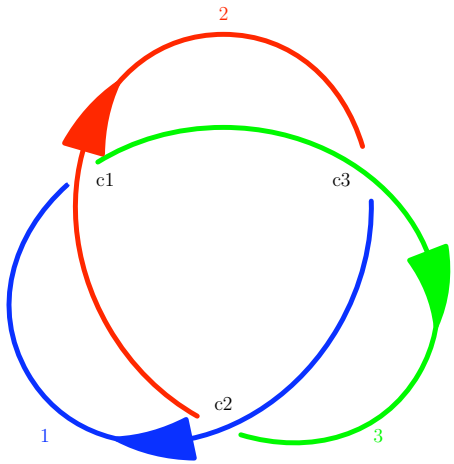
Computing the Alexander polynomial of the link

For a link with $K = 1$ knot:

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

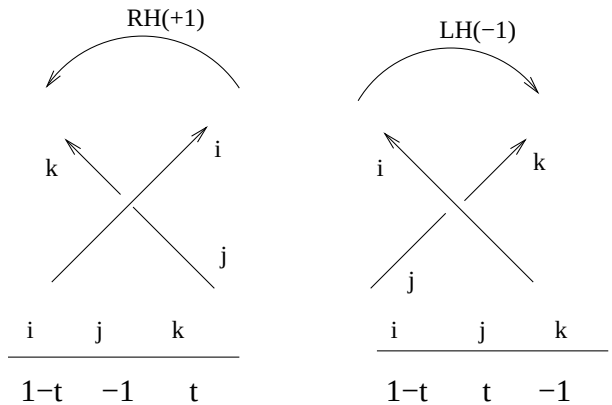
$$D := det(minor(P(L))) = -t^2 + t - 1$$

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



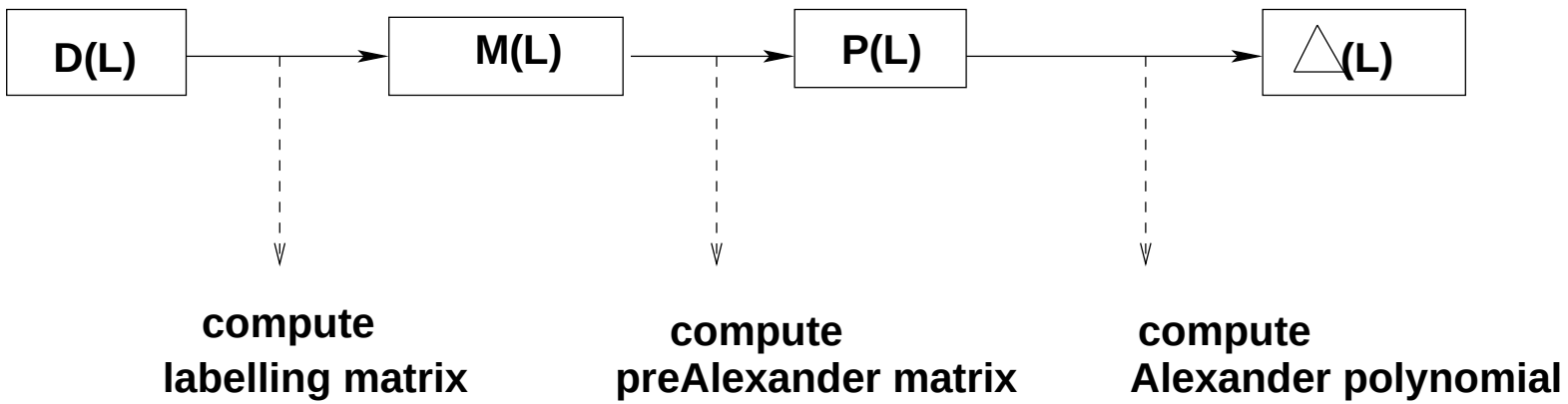
For a link with $K > 1$ knots and n crossings $\Delta(L)$ is the gcd of all the $(n-1) \times (n-1)$ minor determinants of $P(L)$.

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



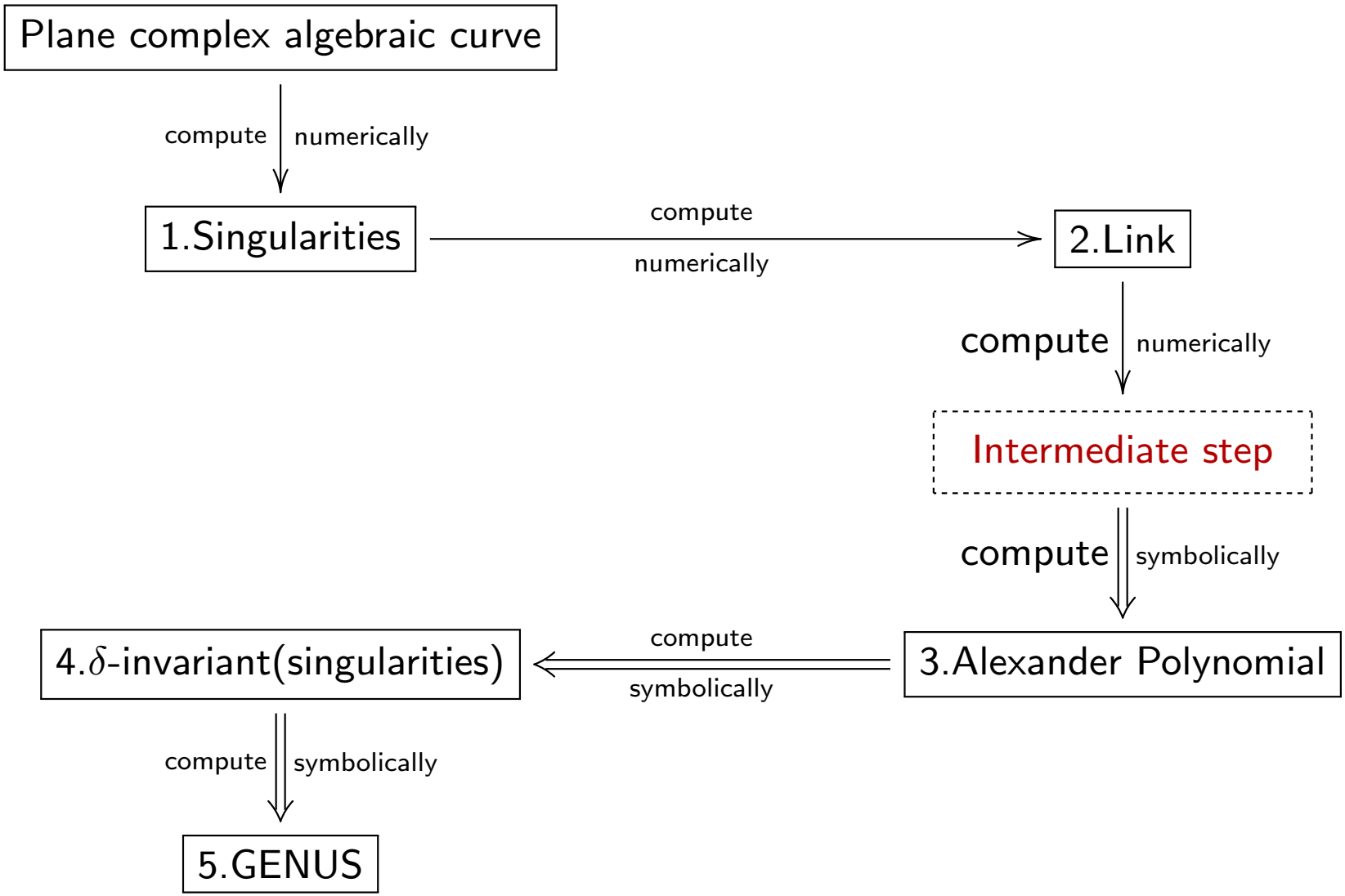
Computing the Alexander polynomial of the link

So, the Alexander polynomial is computed in several steps:

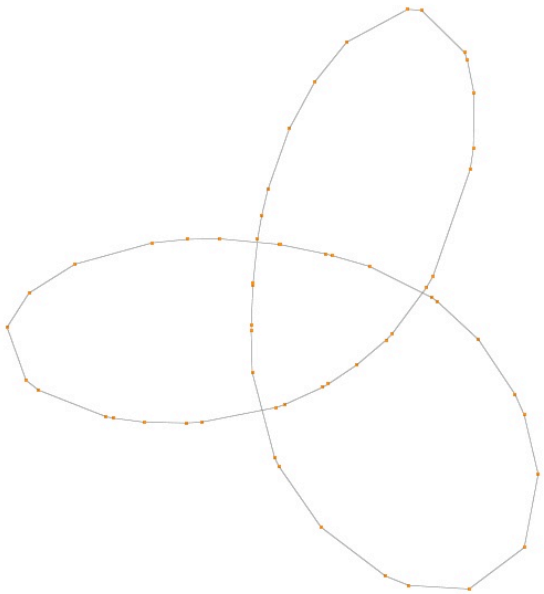


In order to compute it, we need D(L)!

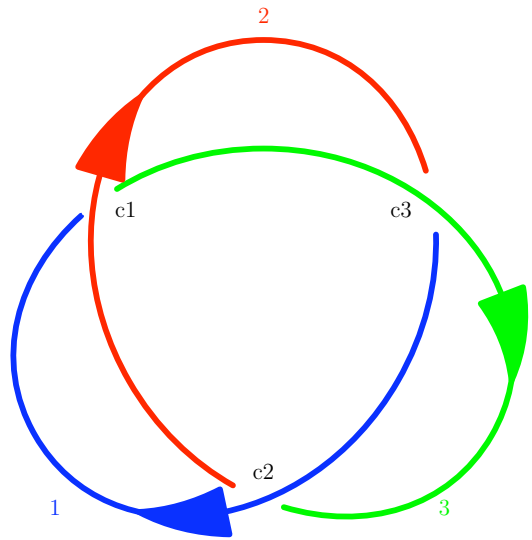
Next


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



$\overset{?}{\Rightarrow}$



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

  e(indexS, indexD)

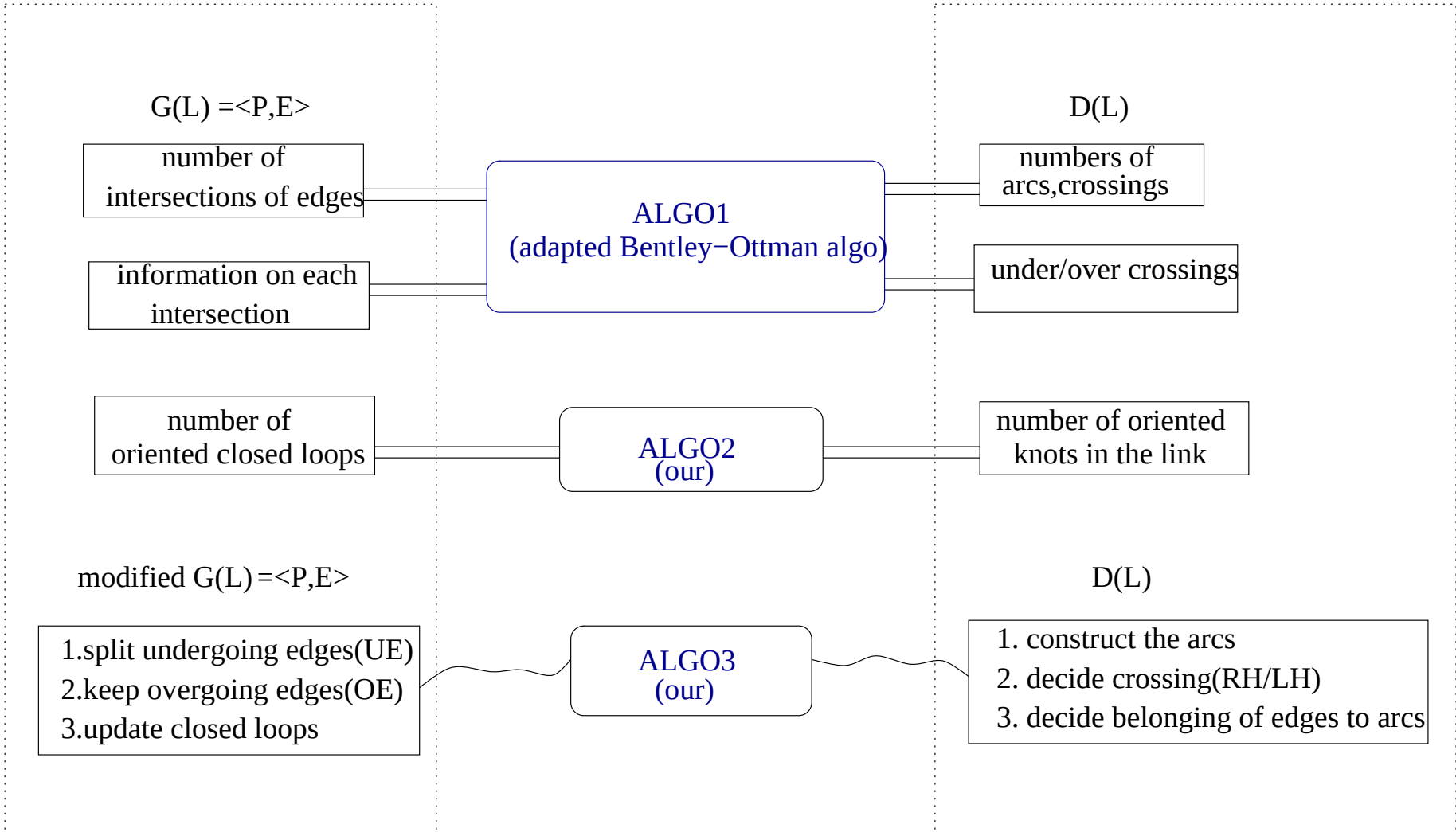
$D(L)$

$\longrightarrow$ number of arcs, crossings

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

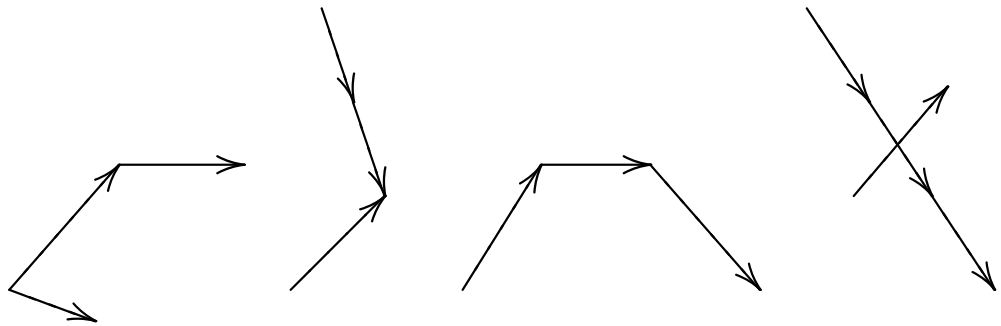
$\longrightarrow$ number of knots in the link(orientation)

Intermediate step



Algorithm 1 - Adapted version of Bentley-Ottman

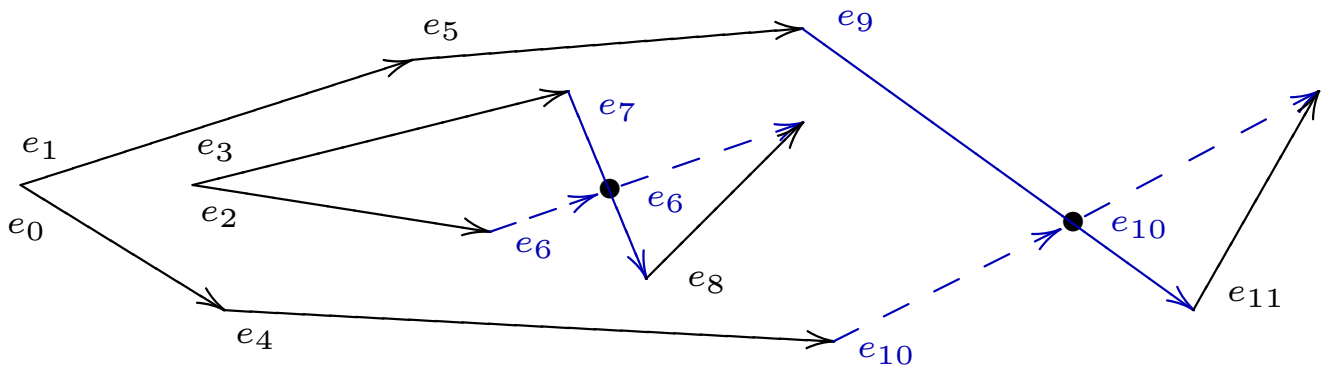
- Input: S a set of "short" edges ordered from left to right and by (1), (2), (3):



- Output: I - the set of all intersections among edges of S and
  - for each $p = e_i \cap e_j \in I$, the "arranged" pair of edges (e_i, e_j), i.e e_i is below e_j in $\mathbb{R}^3$

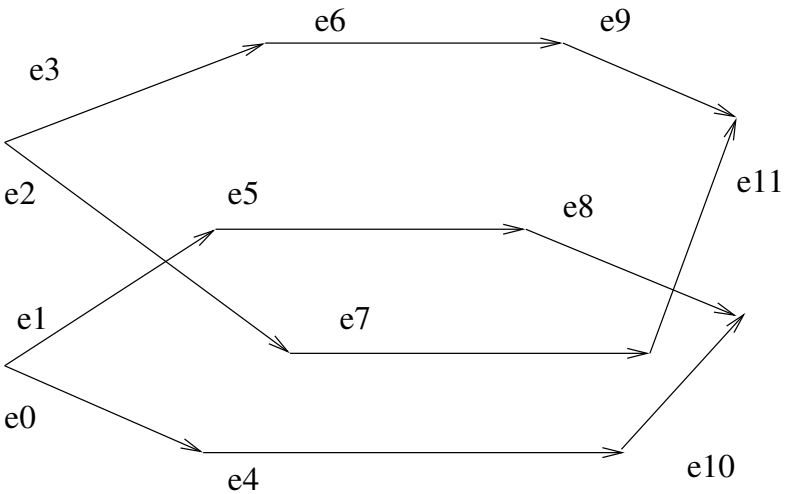
Algorithm 1 - Adapted version of Bentley-Ottman

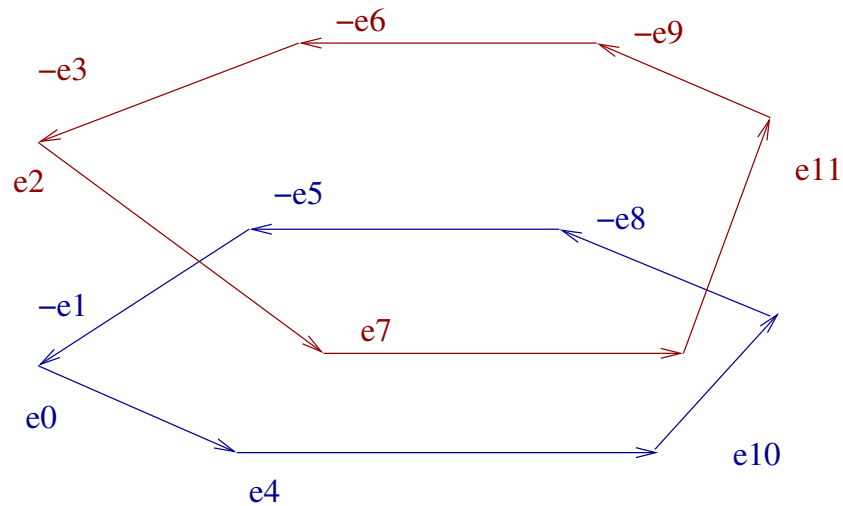
- For instance, on this graph:



- the adapted Bentley-Ottman algorithm produces the 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

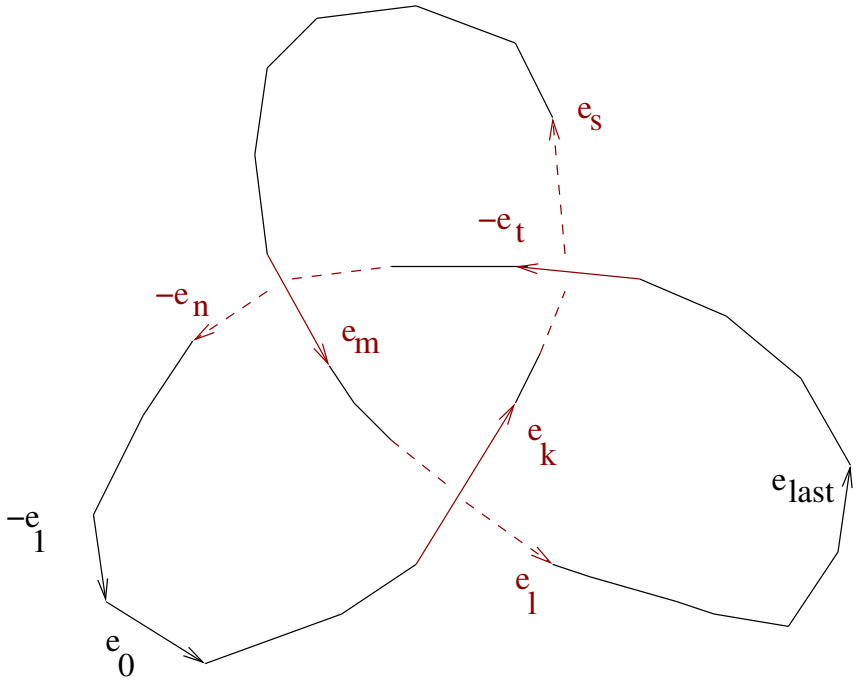


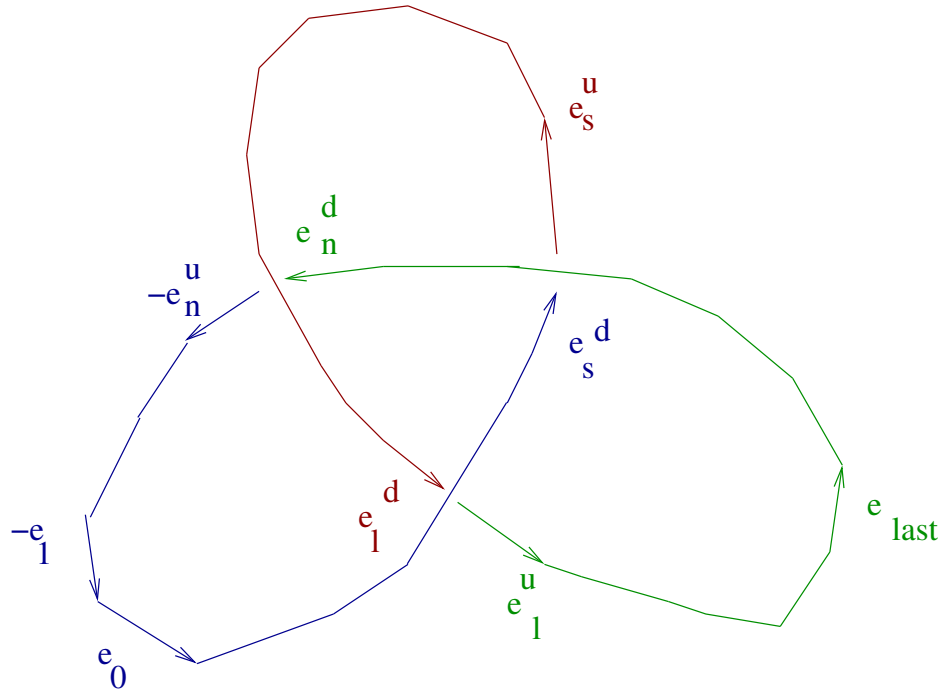


- E ordered by (1),(2),(3) $\Rightarrow$

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

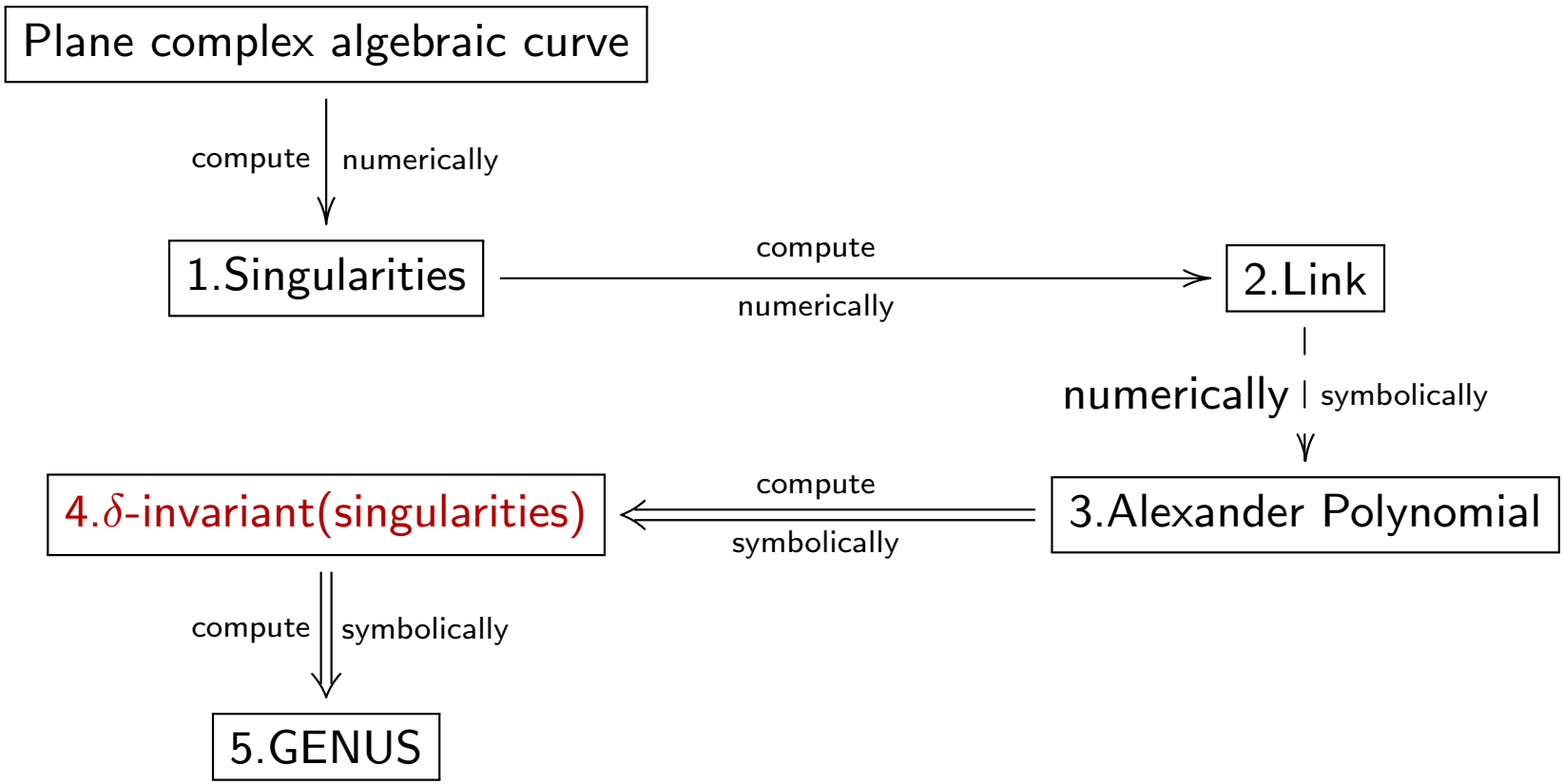
Algorithm 3 - Constructing the arcs





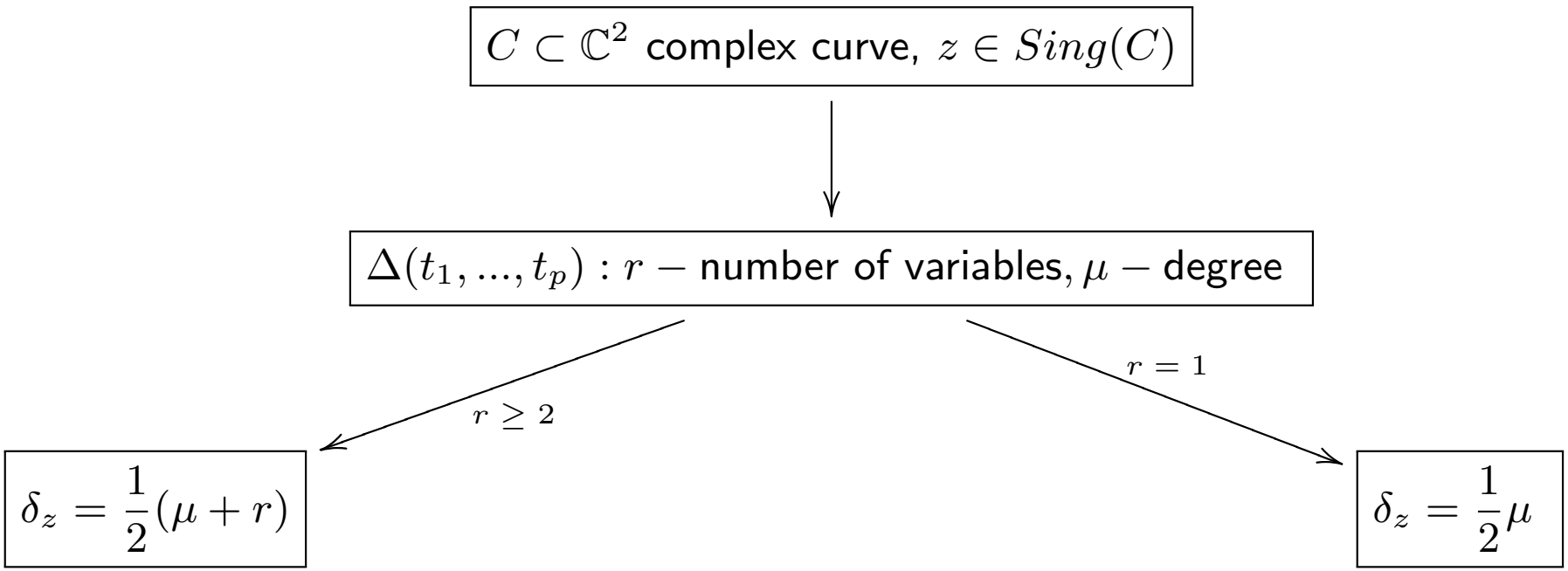
- $E = \{e_0, ..., e_{last}\}$ $\qquad a_0 = \{e_n^u, .., -e_1, e_0, .., e_k, .., e_s^d\}$
- $E_I = \{(-e_n, e_m), (e_l, e_k), (e_s, -e_t)\}$ $\qquad \Rightarrow \qquad a_1 = \{e_l^u, .., -e_t, .., -e_n^d\}$
- $L_0 = \{e_0, ..., e_s, e_l, ..., -e_1\}$ $\qquad a_2 = \{e_s^u, ..., e_m, ..., e_l^d\}$
- while constructing the arcs we also decide the type of crossings (RH or 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

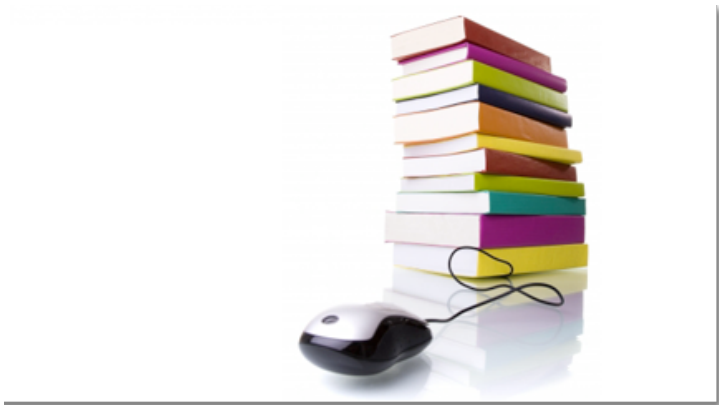
From the Alexander polynomial, we derive the formulae for the δ-invariant:
(based on Milnor's research)



Summary

At present:

- We have a symbolic-numeric algorithm, i.e. an approximate algorithm, for GENus cOMputation of plane Complex algebraiC Curves using Knot theory implemented in the GENOM3CK library (originally QComplexInvariants).

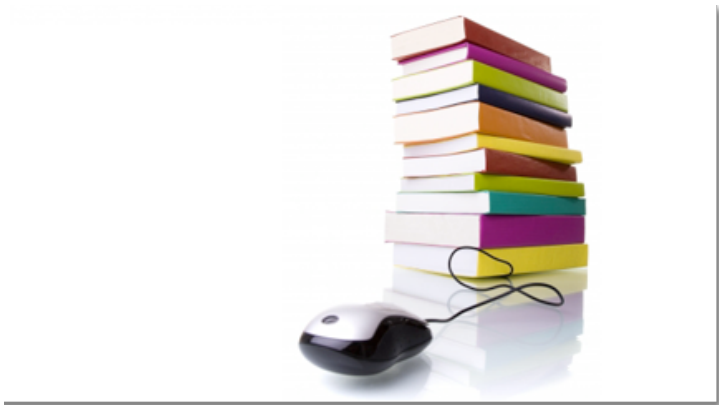


- The algorithm is based on combinatorial techniques from knot theory, that allow us to analyze the singularities of the input curve and to compute the invariants: topology of singularities (algebraic link), Alexander polynomial, δ-invariant, genus. The algorithm depends on the parameter $\epsilon \in \mathbb{R}_+^*$.

Summary

At present:

- We have a symbolic-numeric algorithm, i.e. an approximate algorithm, for GENus cOMputation of plane Complex algebraiC Curves using Knot theory implemented in the GENOM3CK library (originally QComplexInvariantsPlugin).



- Why Axel? (and the mmx libraries)
  - Axel is the only system to compute the topology of smooth implicitly defined curves in $\mathbb{R}^3$ and information on the topology. Thus for our purpose, it offers a major advantage over other systems allowing us to implement the complete symbolic-numeric method for genus computation.

Summary

At present , using the library in Axel we get:

Equation	Link	Alex poly, δ-invariant, genus
$x^2 - y^2, \epsilon = 1.0$	Hopf link	$\Delta(t_1) = 1,\ \delta = 1, g = -1$
$x^2 - y^3, \epsilon = 1.0$	Trefoil knot	$\Delta(t_1) = t_1^2 - t_1 + 1, \delta = 1, g = 0$
$x^2 - y^4, \epsilon = 1.0$	2-knots link	$\Delta(t_1, t_2) = t_1 t_2 + 1, \delta = 2, g = -1$
$x^2 - y^5, \epsilon = 1.0$	1-knot link	$\Delta(t_1) = t_1^4 - t_1^3 + t_1^2 - t_1 + 1, \delta = 2, g = 0$
$x^3 - y^3, \epsilon = 1.0$	3-knots link	$\Delta(t_1, t_2, t_3) = -t_1 t_2 t_3 + 1, \delta = 3, g = -2$
$x^4 + x^2 y + y^5, \epsilon = 0.5$	3-knots link	$\Delta(t_1, t_2, t_3) = -t_1^2 t_2^2 t_3 + 1, \delta = 4, g = 2$

Summary

Next:

- For an arbitrary plane complex algebraic curve C defined by a polynomial with coefficients of limited accuracy, i.e $F(x, y) = -x^3 - 1.875xy + y^2 - 0.0xy$, we want to compute the approximate $genus(C)$ using GENOM3CK.
  Important questions arise:

Summary

Next:

- For an arbitrary plane complex algebraic curve C defined by a polynomial with coefficients of limited accuracy, i.e $F(x, y) = -x^3 - 1.875xy + y^2 - 0.0xy$, we want to compute the approximate $genus(C)$ using GENOM3CK.
  Important questions arise:
  - What does one mean by approximate genus?
  - How does one control the error in numerical computation?

Preliminaries-Approximate algebraic computation

Objects of approximate algebraic computation[1]: polynomials with coefficients of limited accuracy, i.e. $F(x,y) = F(x,y) = -x^3 - 1.875xy + y^2 - 0.0xy$.

Basic questions

What happens when using approximate computation?

Why using approximate computation?

What is (one) of the aims of approximate computation?

[1] Thanks to the colleagues from the DK for their helpful discussions

Preliminaries-Approximate algebraic computation

Objects of approximate algebraic computation[1]: polynomials with coefficients of limited accuracy, i.e. $F(x,y) = F(x,y) = -x^3 - 1.875xy + y^2 - 0.0xy$.

Basic questions

What happens when using approximate computation?

Why using approximate computation?

What is (one) of the aims of approximate computation?

Tiny perturbations in data input produce huge error in solution (ill-posed problems). We get failure of classical algorithms: Euclidean algorithm, root polynomial computation, genus computation, etc.

Definition (Hadamard). A problem is well posed if: it has a solution, the solution is unique, and the solution depends continuously on data and parameters.
Remark. If the solution of the problem depends in a discontinuous way on the data, then small errors can create large deviations, and the problem is called ill-posed.

[1] Thanks to the colleagues from the DK for their helpful discussions

Preliminaries-Approximate algebraic computation

Objects of approximate algebraic computation[1]: polynomials with coefficients of limited accuracy, i.e. $F(x,y) = F(x,y) = -x^3 - 1.875xy + y^2 - 0.0xy$.

Basic questions

What happens when using approximate computation?

Why using approximate computation?

What is (one) of the aims of approximate computation?

There is no other choice since the input data are only approximately known!

[1] Thanks to the colleagues from the DK for their helpful discussions

Preliminaries-Approximate algebraic computation

Objects of approximate algebraic computation[1]: polynomials with coefficients of limited accuracy, i.e. $F(x,y) = F(x,y) = -x^3 - 1.875xy + y^2 - 0.0xy$.

Basic questions

What happens when using approximate computation?
Why using approximate computation?
What is (one) of the aims of approximate computation?

To deal with ill-posed problems in numerical computation!

What should a numerical algorithm really do?
$\Rightarrow$ Naive answer: Compute solutions.
$\Rightarrow$ Z. Zeng, E. Kaltofen, H. Stetter: A numerical algorithm generates the exact solution of a nearby problem (related with regularization theory).

[1] Thanks to the colleagues from the DK for their helpful discussions

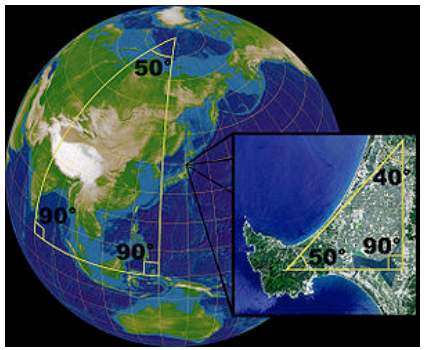
Genus computation - Approximate algebraic computation

Approximate algebraic computation to an ill-posed problem

- 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.

What is a manifold and its dimension?

- A manifold M is a topological space that is locally euclidean to R^n, n is the dimension of M. (any object that can be "charted" is a manifold.)



Genus computation - Approximate algebraic computation

Approximate algebraic computation to an ill-posed problem

- We partition the data input of the problem into pejorative manifolds. For given input we need to determine the nearby pejorative manifold of the highest codimension (i.e the smallest nearby pejorative manifold).

What does "nearby" means?

- "Nearby" depends on the input parameter ϵ.
- It is not precise what "nearby" means.

GENOM3CK and approximate algebraic computation

We consider the exact algorithm for genus computation as the function:

$$E : \mathbb{C}[x, y] \to \mathbb{Z}, F(x, y) \mapsto E(F(x, y)).$$

We consider the approximate algorithm from GENOM3CK for genus computation as the function:

$$A_\epsilon : \mathbb{C}[x, y] \times \mathbb{R}_+^* \to \mathbb{Z}, F(x, y) \mapsto A_\epsilon(F(x, y)).$$

Remark: The output of A_ϵ: the Alexander polynomial (Δ), the δ-invariant (δ), and the genus (g).

GENOM3CK and approximate algebraic computation

Tests experiments performed with GENOM3CK indicate two important properties of A_ϵ:

Convergency

- we consider $F(x,y)$ with both exact and inexact coefficients; we compute $A_\epsilon(F(x,y))$ for different values of the parameter ϵ.
- for $-x^3 - xy + y^2$, we know that the exact topology is the Hopf link;
- we notice that the approximate solution computed with A_ϵ converges to the exact solution as ϵ tends to 0: $\underset{F(x,y)}{\forall} \lim_{\epsilon \to 0} A_\epsilon(F(x,y)) = E(F(x,y))$.

Equation and ϵ		**Link**	**Alexander, δ invariants, genus**		
$-x^3 - xy + y^2$	1.00	Trefoil knot	$\Delta(t_1) = t_1^2 - t_1 + 1$	$\delta = 1$	$g = 0$
$-x^3 - xy + y^2$	0.5	Trefoil knot	$\Delta(t_1) = t_1^2 - t_1 + 1$	$\delta = 1$	$g = 0$
$-x^3 - xy + y^2$	0.25	Hopf link	$\Delta(t_1, t_2) = 1$	$\delta = 1$	$g = 0$
$-x^3 - xy + y^2$	0.14	Hopf link	$\Delta(t_1, t_2) = 1$	$\delta = 1$	$g = 0$

GENOM3CK and approximate algebraic computation

Tests experiments performed with GENOM3CK indicate two properties of A_ϵ:

Continuity

- we consider $p(x, y)$ a polynomial with exact coefficients;
- for $\delta \in \mathbb{R}$ we consider $p_\delta(x, y)$ perturbations of p;
- perturbations of type I: $p_\delta(x, y) = p(x, y) + \delta$, where $\delta \in \mathbb{R}^*$.
- perturbations of type II: $p_\delta(x, y) = p(x, y) + \delta q(x, y)$, where $\delta \in \mathbb{R}^*$, $q(x, y) \in \mathbb{C}[x, y]$ is an arbitrary exact polynomial.
- we consider $F(x, y) := p_\sigma(x, y)$, and several values for ϵ. For each ϵ, we compute $A_\epsilon(F(x, y))$ for different values of δ.
- we observe that small changes on the input data produce small changes on the output solution:

$$\underset{F(x,y)}{\forall}\ \underset{\eta>0}{\exists} \text{ such that } \underset{\epsilon<\eta}{\forall}\ \underset{\eta_1>0}{\exists}\ \underset{G(x,y)}{\forall} G(x, y) \in I := (F(x, y) - \eta_1, F(x, y) + \eta_1)$$
$$A_\epsilon(G(x, y)) \text{ is constant in } I.$$

GENOM3CK and approximate algebraic computation

Continuity (next) small changes in the input produce small changes in the output:

Perturbations I and ϵ		$\sigma = 10^{-e}, e \in \mathbb{N}^*$	Link	Invariants
$-x^3 - xy + y^2 - 10^{-e}$	0.5	$\{10^{-2}, ..., 10^{-10}\}$	Trefoil knot	$\Delta(t_1) = t_1^2 - t_1 + 1$ $\delta = 1$ $g = 0$
$-x^3 - xy + y^2 - 10^{-e}$	0.25	$\{10^{-2}, ..., 10^{-10}\}$	Hopf link	$\Delta(t_1, t_2) = 1$ $\delta = 1$ $g = 0$

$p(x, y) = -x^3 - xy + y^2$ $q(x, y) = -x^3 - 2xy + y^2$;
$F(x, y) := p_\delta(x, y) = p(x, y) + \delta q(x, y) = -(1+10^{-e})x^3 - (1+2\cdot 10^{-e})xy + (1+10^{-e})y^2$
$\delta = 0.1 : F(x, y) = -1.1x^3 - 1.2x^2 + 1.1y^2$
$\delta = 0.01 : F(x, y) = -1.01x^3 - 1.02x^2 + 1.01y^2$, etc .

Perturbations II and ϵ		$\sigma = 10^{-e}, e \in \mathbb{N}^*$	Link	Invariants
$-(1+10^{-e})x^3 - (1+2\cdot 10^{-e})xy + (1+10^{-e})y^2$	0.15	$\{10^{-1}, ..., 10^{-10}\}$	Hopf link	$\Delta(t_1, t_2) = 1$ $\delta = 1$ $g = 0$
$-(1+10^{-e})x^3 - (1+2\cdot 10^{-e})xy + (1+10^{-e})y^2$	0.14	$\{10^{-1}, ..., 10^{-10}\}$	Hopf link	$\Delta(t_1, t_2) = 1$ $\delta = 1$ $g = 0$

GENOM3CK and approximate algebraic computation

- For an arbitrary plane complex algebraic curve C defined by the polynomial $F(x, y) = -x^3 - 1.875xy + y^2 - 0.0xy$ we compute $genus(C)$ using the approximate algorithm $A_\epsilon(x, y)$.
- A_ϵ computes $genus(C) = 0$. Since A_ϵ is continuous and convergent:
  - There is no nearby polynomial with genus less than 0;
  - There is a nearby polynomial with genus exactly 0;
- What does nearby means?

GENOM3CK and approximate algebraic computation

- For an arbitrary plane complex algebraic curve C defined by the polynomial $F(x,y) = -x^3 - 1.875xy + y^2 - 0.0xy$ we compute $genus(C)$ using the approximate algorithm $A_\epsilon(x,y)$.
- A_ϵ computes $genus(C) = 0$. Since A_ϵ is continuous and convergent:
  - There is no nearby polynomial with genus less than 0;
    $$\underset{F(x,y)}{\forall} \underset{G(x,y)}{\forall} |F(x,y) - G(x,y)| < \epsilon_1 \Rightarrow E(G(x,y)) \leq A_{\epsilon_1}(F(x,y))$$
  - There is a nearby polynomial with genus exactly 0;
    $$\underset{F(x,y)}{\forall} \underset{H(x,y)}{\exists} |F(x,y) - H(x,y)| < \epsilon_2 \Rightarrow E(H(x,y)) = A_{\epsilon_2}(F(x,y))$$

GENOM3CK and approximate algebraic computation

- For an arbitrary plane complex algebraic curve C defined by the polynomial $F(x,y) = -x^3 - 1.875xy + y^2 - 0.0xy$ we compute $genus(C)$ using the approximate algorithm $A_\epsilon(x,y)$.
- A_ϵ computes $genus(C) = 0$. Since A_ϵ is continuous and convergent:
  - There is no nearby polynomial with genus less than 0;
    $$\underset{F(x,y)}{\forall} \underset{G(x,y)}{\forall} |F(x,y) - G(x,y)| < d_1(\epsilon) \Rightarrow E(G(x,y)) \leq A_\epsilon(F(x,y))$$
  - There is a nearby polynomial with genus exactly 0;
    $$\underset{F(x,y)}{\forall} \underset{H(x,y)}{\exists} |F(x,y) - H(x,y)| < d_2(\epsilon) \Rightarrow E(H(x,y)) = A_\epsilon(F(x,y))$$
- $d_1, d_2 : \mathbb{R}_+ \to \mathbb{R}_+$ continuous, bijective and increasing functions.
- the computed $genus(C)$ is in fact approximate $genus(C)$,
  i.e. the lowest possible genus of a curve defined by a "nearby" polynomial

Conclusion and future work

Achieved goals:

- complete automatization for the steps of the approximate algorithm (in GENOM3CK); invariants as algebraic link, Alexander polynomial, delta-invariant, genus, are computed;
- tests experiments show that the approximate algorithm has the continuity and convergency properties;

TO DO's:

Conclusion and future work

Achieved goals:

- complete automatization for the steps of the approximate algorithm (in GENOM3CK); invariants as algebraic link, Alexander polynomial, delta-invariant, genus, are computed;
- tests experiments show that the approximate algorithm has the continuity and convergency properties;

TO DO's:

- prove the properties of the approximate algorithm (i.e. continuity, convergency);

Conclusion and future work

Achieved goals:

- complete automatization for the steps of the approximate algorithm (in GENOM3CK); invariants as algebraic link, Alexander polynomial, delta-invariant, genus, are computed;
- tests experiments show that the approximate algorithm has the continuity and convergency properties;
- the approximate algorithm computes discrete information from continuous data; it can be described using principles from regularization theory and approximate algebraic computation.

TO DO's:

- prove the properties of the approximate algorithm (i.e. continuity, convergency);

Conclusion and future work

Achieved goals:

- complete automatization for the steps of the approximate algorithm (in GENOM3CK); invariants as algebraic link, Alexander polynomial, delta-invariant, genus, are computed;
- tests experiments show that the approximate algorithm has the continuity and convergency properties;
- the approximate algorithm computes discrete information from continuous data; it can be described using principles from regularization theory and approximate algebraic computation.

TO DO's:

- prove the properties of the approximate algorithm (i.e. continuity, convergency);
- make precise the meaning of the computed approximate output with the approximate algorithm.

Thank you for your attention.
Questions?