Structured algorithms for algebraic curves

Simon Abelard

Laboratoire Cryptologie et Composants
Thales SIX GTS France

Two research axes

Point counting:

Cryptographic applications:

- Constructive: discrete log.
- Destructive: VDF (verifiable-delay functions)

Riemann-Roch spaces:

Applications:

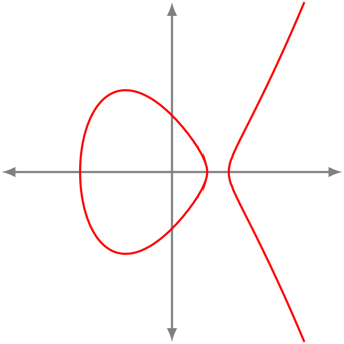
- Symbolic integration
- Arithmetic in Jacobians
- Algebraic Geometry codes

Common Denominator:

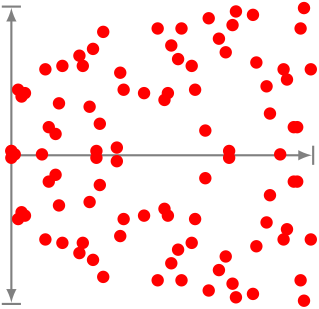
- Algebraic curves
- Protection of information
- Computer Algebra
- **Structured problems**

Part I : hyperelliptic point counting

Input: hyperelliptic curve $y^2 = f(x)$ over $\mathbb{F}_p$.
Problem: how many solutions of $y^2 = f(x)$ mod p ?



$y^2 = x^3 - 2x + 1$ over $\mathbb{R}$



$y^2 = x^3 - 2x + 1$ over $\mathbb{F}_{89}$

Hyperelliptic point counting

Example: $y^2 = x^7 - 7x^5 + 14x^3 - 7x + 42$ over $\mathbb{F}_{2^{64}-59}$.
Parameters: p, degree of f denoted $2g + 1$ (g is the **genus**).

Hyperelliptic point counting

Example: $y^2 = x^7 - 7x^5 + 14x^3 - 7x + 42$ over $\mathbb{F}_{2^{64}-59}$.
Parameters: p, degree of f denoted $2g+1$ (g is the **genus**).

Polynomial-time algorithms

Input size: $O(g \log p)$.
Algorithm polynomial in $g \log p$? $\rightsquigarrow$ open problem.

Hyperelliptic point counting

Example: $y^2 = x^7 - 7x^5 + 14x^3 - 7x + 42$ over $\mathbb{F}_{2^{64}-59}$.
Parameters: p, degree of f denoted $2g+1$ (g is the **genus**).

Polynomial-time algorithms

Input size: $O(g \log p)$.
Algorithm polynomial in $g \log p$? $\rightsquigarrow$ open problem.

- p-adic approaches are polynomial in g. (i.e. $(pg)^{O(1)}$)
- ℓ-adic approaches are polynomial in $\log p$. (i.e. $(\log p)^{e(g)}$)

Hyperelliptic point counting

Example: $y^2 = x^7 - 7x^5 + 14x^3 - 7x + 42$ over $\mathbb{F}_{2^{64}-59}$.
Parameters: p, degree of f denoted $2g+1$ (g is the **genus**).

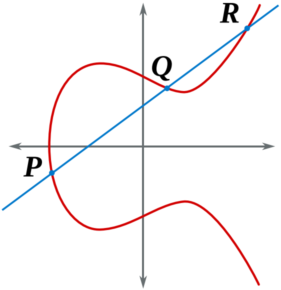
Polynomial-time algorithms

Input size: $O(g \log p)$.
Algorithm polynomial in $g \log p$? $\rightsquigarrow$ open problem.

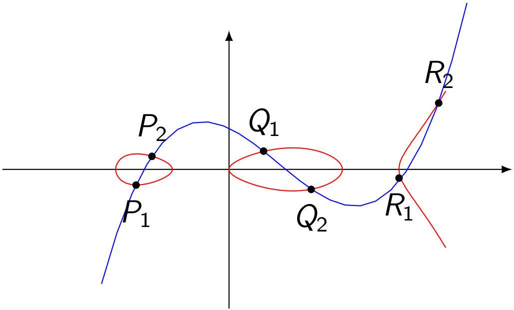
- p-adic approaches are polynomial in g. (i.e. $(pg)^{O(1)}$)
- ℓ-adic approaches are polynomial in $\log p$. (i.e. $(\log p)^{e(g)}$)

Our contributions: large p, exponent of $\log p$ depends on g.

From curves to groups



$\boldsymbol{P + Q + R = 0}$



$$P_1 + P_2 + Q_1 + Q_2 + R_1 + R_2 = 0$$

Curve of equation $Y^2 = X^5 - 2X^4 - 7X^3 + 8X^2 + 12X$

$J = \mathrm{Jac}(\mathcal{C})$ is the Jacobian, its elements are formal sums of points.

From ℓ-adic methods to polynomial systems

Let $\mathcal{C} : y^2 = f(x)$ be a hyperelliptic curve over $\mathbb{F}_q$.
Let J be its Jacobian and g its genus. We want $N = \#J(\mathbb{F}_q)$.

1. (Hasse-Weil) bounds on N $\Rightarrow$ compute N mod ℓ
2. ℓ-torsion $J[\ell] = \{D \in J | \ell D = 0\} \simeq (\mathbb{Z}/\ell\mathbb{Z})^{2g}$
3. Action of Frobenius $\pi : (x, y) \mapsto (x^q, y^q)$ on $J[\ell]$ yields N mod ℓ

Algorithm *a la* Schoof

For sufficiently many primes ℓ

Describe I_ℓ the ideal of ℓ-torsion

Compute action of π on $J[\ell]$

Deduce N mod ℓ

Recover N by CRT

From ℓ-adic methods to polynomial systems

Let $\mathcal{C} : y^2 = f(x)$ be a hyperelliptic curve over $\mathbb{F}_q$.
Let J be its Jacobian and g its genus. We want $N = \#J(\mathbb{F}_q)$.

1. (Hasse-Weil) bounds on N $\Rightarrow$ compute N mod ℓ
2. ℓ-torsion $J[\ell] = \{D \in J | \ell D = 0\} \simeq (\mathbb{Z}/\ell\mathbb{Z})^{2g}$
3. Action of Frobenius $\pi : (x, y) \mapsto (x^q, y^q)$ on $J[\ell]$ yields N mod ℓ

Algorithm *a la* Schoof

For sufficiently many primes ℓ
- Describe I_ℓ the ideal of ℓ-torsion
- Compute action of π on $J[\ell]$
- Deduce N mod ℓ

Recover N by CRT

From ℓ-adic methods to polynomial systems

Let $\mathcal{C} : y^2 = f(x)$ be a hyperelliptic curve over $\mathbb{F}_q$.
Let J be its Jacobian and g its genus. We want $N = \#J(\mathbb{F}_q)$.

1. (Hasse-Weil) bounds on N $\Rightarrow$ compute $N \bmod \ell$
2. ℓ-torsion $J[\ell] = \{D \in J | \ell D = 0\} \simeq (\mathbb{Z}/\ell\mathbb{Z})^{2g}$
3. Action of Frobenius $\pi : (x, y) \mapsto (x^q, y^q)$ on $J[\ell]$ yields $N \bmod \ell$

Algorithm *a la* Schoof

For sufficiently many primes ℓ

  Describe I_ℓ the ideal of ℓ-torsion

  Compute action of π on $J[\ell]$

  Deduce $N \bmod \ell$

Recover N by CRT

From ℓ-adic methods to polynomial systems

Let $\mathcal{C} : y^2 = f(x)$ be a hyperelliptic curve over $\mathbb{F}_q$.
Let J be its Jacobian and g its genus. We want $N = \#J(\mathbb{F}_q)$.

1. (Hasse-Weil) bounds on N $\Rightarrow$ compute N mod ℓ
2. ℓ-torsion $J[\ell] = \{D \in J | \ell D = 0\} \simeq (\mathbb{Z}/\ell\mathbb{Z})^{2g}$
3. Action of Frobenius $\pi : (x, y) \mapsto (x^q, y^q)$ on $J[\ell]$ yields N mod ℓ

Algorithm *a la* Schoof

For sufficiently many primes ℓ

- Describe I_ℓ the ideal of ℓ-torsion
- Compute action of π on $J[\ell]$
- Deduce N mod ℓ

Recover N by CRT

Main tasks: find equations for I_ℓ (and bound their degree).
Solve these equations (i.e. find a Gröbner basis for I_ℓ).

Contribution I, genus-3 curves

Asymptotic complexities

Genus	Complexity	Authors
$g = 1$	$\widetilde{O}(\log^4 p)$	Schoof, Elkies, Atkin ($\sim$1990)
$g = 2$	$\widetilde{O}(\log^8 p)$	Gaudry, Schost (2000)
$g = 2$, RM curves	$\widetilde{O}(\log^5 p)$	Gaudry, Kohel, Smith (2011)
$g = 3$	$\widetilde{O}(\log^{14} p)$	Our work[1]
$g = 3$, RM curves	$\widetilde{O}(\log^6 p)$	Our work[1]

Practical experiment[1]

Curve $y^2 = x^7 - 7x^5 + 14x^3 - 7x + 42$ over $\mathbb{F}_{2^{64}-59}$.
Record computation : 64-bit p, Jacobian has order $\sim 2^{192}$.

[1] A., Gaudry, Spaenlehauer. Proceedings of ANTS 2018

Applications: Verifiable Delay Functions (VDF)

Function f such that:

- Evaluation $x \mapsto f(x)$ slow and sequential (hard to parallelize).
- Verifying that $y = f(x)$ is fast.

Use of VDFs: randomness, power-saving blockchains.

Construction: $f(\gamma) = \gamma^{2^T}$, γ in a group of unknown order.

Applications: Verifiable Delay Functions (VDF)

Function f such that:

- Evaluation $x \mapsto f(x)$ slow and sequential (hard to parallelize).
- Verifying that $y = f(x)$ is fast.

Use of VDFs: randomness, power-saving blockchains.

Construction: $f(\gamma) = \gamma^{2^T}$, γ in a group of unknown order.

Use case: the group is the **Jacobian of a curve**.
Point counting $\rightsquigarrow$ order of the Jacobian, must be infeasible.
Improvement on point counting $\rightsquigarrow$ threatens security.

Applications: Verifiable Delay Functions (VDF)

Function f such that:

- Evaluation $x \mapsto f(x)$ slow and sequential (hard to parallelize).
- Verifying that $y = f(x)$ is fast.

Use of VDFs: randomness, power-saving blockchains.

Construction: $f(\gamma) = \gamma^{2^T}$, γ in a group of unknown order.

Use case: the group is the **Jacobian of a curve**.
Point counting $\rightsquigarrow$ order of the Jacobian, must be infeasible.
Improvement on point counting $\rightsquigarrow$ threatens security.

Impact of our work:

- Choosing safe parameters (p large enough)
- Avoid certain weaker curves

Point-counting in genus 3

Remember: our problem boils down to a polynomial system.

Point-counting in genus 3

Remember: our problem boils down to a polynomial system.

In theory:

- 6 equations of degree $O(\ell^2)$
- Solved using trivariate resultants (good when few variables, good complexity results)
- Final complexity: $\widetilde{O}(\log^{14} q)$

Point-counting in genus 3

Remember: our problem boils down to a polynomial system.

In theory:

- 6 equations of degree $O(\ell^2)$
- Solved using trivariate resultants (good when few variables, good complexity results)
- Final complexity: $\widetilde{O}(\log^{14} q)$

In practice for $\ell = 3$:

- 5 equations and variables degrees ≤ 55
- Solved using Gröbner bases (F4 in Magma): apparently nice structure but no proven complexity bounds
- Runs in 2 weeks using 140 GB of RAM
- $\ell = 5$ is out of reach in practice

Point-counting in genus 3

Remember: our problem boils down to a polynomial system.

In theory:

- 6 equations of degree $O(\ell^2)$
- Solved using trivariate resultants (good when few variables, good complexity results)
- Final complexity: $\widetilde{O}(\log^{14} q)$

In practice for $\ell = 3$:

- 5 equations and variables degrees ≤ 55
- Solved using Gröbner bases (F4 in Magma): apparently nice structure but no proven complexity bounds
- Runs in 2 weeks using 140 GB of RAM
- $\ell = 5$ is out of reach in practice

Culprit: size of ℓ-torsion (ℓ^6 in genus 3).
$\Rightarrow$ Look for more favorable curves.

Tuning Schoof's algorithm using RM

An RM family (Mestre'91,Tautz-Top-Verberkmoes'91)

Family $\mathcal{C}_t : y^2 = x^7 - 7x^5 + 14x^3 - 7x + t$ with $t \neq \pm 2$.
$\longrightarrow$ hyperelliptic curves of genus 3.
Set η_7 root of $X^3 + X^2 - 2X - 1$, $\mathbb{Z}[\eta_7] \subset \mathrm{End}(\mathrm{Jac}(\mathcal{C}_t))$.

Tuning Schoof's algorithm using RM

An RM family (Mestre'91,Tautz-Top-Verberkmoes'91)

Family $\mathcal{C}_t : y^2 = x^7 - 7x^5 + 14x^3 - 7x + t$ with $t \neq \pm 2$.
$\longrightarrow$ hyperelliptic curves of genus 3.
Set η_7 root of $X^3 + X^2 - 2X - 1$, $\mathbb{Z}[\eta_7] \subset \mathrm{End}(\mathrm{Jac}(\mathcal{C}_t))$.

Example: $(13) = (2 - \eta_7 - 2\eta_7^2)(-2 + 2\eta_7 + \eta_7^2)(3 + \eta_7 - \eta_7^2)$.

The 13-torsion is direct sum of three kernels of endomorphisms.
We model these kernels instead $\rightsquigarrow$ 3 systems with:

- 5 variables (like $\ell = 3$ before)
- 5 equations of degrees ≤ 52 (smaller than case $\ell = 3$)

As before, use Gröbner bases in practice (3×3 days and 41 GB).

Tuning Schoof's algorithm using RM

An RM family (Mestre'91,Tautz-Top-Verberkmoes'91)

Family $\mathcal{C}_t : y^2 = x^7 - 7x^5 + 14x^3 - 7x + t$ with $t \neq \pm 2$.
$\longrightarrow$ hyperelliptic curves of genus 3.
Set η_7 root of $X^3 + X^2 - 2X - 1$, $\mathbb{Z}[\eta_7] \subset \mathrm{End}(\mathrm{Jac}(\mathcal{C}_t))$.

Example: $(13) = (2 - \eta_7 - 2\eta_7^2)(-2 + 2\eta_7 + \eta_7^2)(3 + \eta_7 - \eta_7^2)$.

The 13-torsion is direct sum of three kernels of endomorphisms.
We model these kernels instead $\rightsquigarrow$ 3 systems with:

- 5 variables (like $\ell = 3$ before)
- 5 equations of degrees ≤ 52 (smaller than case $\ell = 3$)

As before, use Gröbner bases in practice (3×3 days and 41 GB).

In theory: 3 systems but degrees in $O(\ell^{2/3})$ instead of $O(\ell^2)$.
Final complexity result: $\widetilde{O}((\log q)^6)$ for genus-3 RM hyp. curves.

A practical example

$\mathcal{C}_{42} : y^2 = x^7 - 7x^5 + 14x^3 - 7x + 42$ over $\mathbb{F}_p$ with $p = 2^{64} - 59$.

mod ℓ^k	#var	degree bounds	time	memory
2	—	—	—	—
4 (inert^2)	6	15	1 min	negl.
3 (inert)	5	55	14 days	140 GB
$13 = \mathfrak{p}_1\mathfrak{p}_2\mathfrak{p}_3$	5	52	3×3 days	41 GB

A practical example

$\mathcal{C}_{42} : y^2 = x^7 - 7x^5 + 14x^3 - 7x + 42$ over $\mathbb{F}_p$ with $p = 2^{64} - 59$.

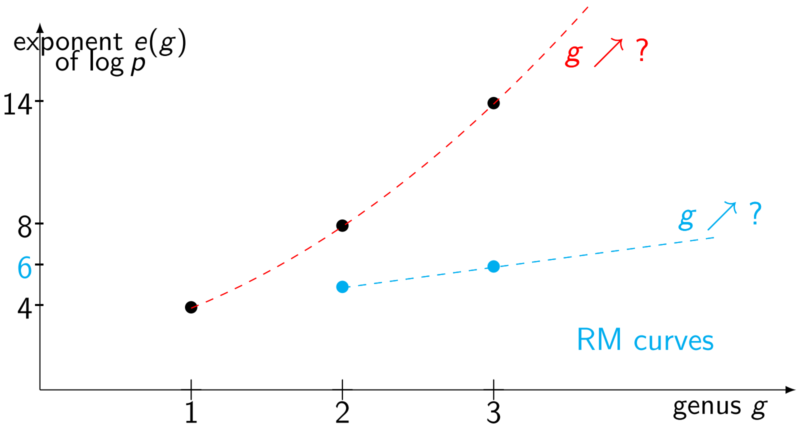
mod ℓ^k	#var	degree bounds	time	memory
2	—	—	—	—
4 (inert^2)	6	15	1 min	negl.
3 (inert)	5	55	14 days	140 GB
$13 = \mathfrak{p}_1\mathfrak{p}_2\mathfrak{p}_3$	5	52	3×3 days	41 GB

Finishing the computation

Like in genus 2, end with exponential collision search.
[Matsuo-Chao-Tsujii'02,Gaudry-Schost'04,Galbraith-Ruprai'09].
Modular info saves factor $156^{3/2} \simeq 1950$.
Cost: 105 CPU-days done in a few hours.

Contribution II, curves of arbitrary genus

With ℓ-adic algorithms : complexity in $c_g(\log p)^{e(g)}$.



Genus	$g=1$	$g=2$	$g=3$
Complexity	$\widetilde{O}(\log^4 p)$	$\widetilde{O}(\log^8 p)$	$\widetilde{O}(\log^{14} p)$

Contribution II, curves of arbitrary genus

Behavior of the exponent when g grows

Adleman-Huang (2001): $e(g) \in O(g^2 \log g)$.

[2] A., Gaudry, Spaenlehauer. Foundations of Comput. Math., 2019

[3] A. Journal of Complexity, 2020

Contribution II, curves of arbitrary genus

Behavior of the exponent when g grows

Adleman-Huang (2001): $e(g) \in O(g^2 \log g)$.

Our work:

- Linear bound for exponent[2].
- Constant exponent in the RM-case[3].

[2] A., Gaudry, Spaenlehauer. Foundations of Comput. Math., 2019

[3] A. Journal of Complexity, 2020

Contribution II, curves of arbitrary genus

Behavior of the exponent when g grows

Adleman-Huang (2001): $e(g) \in O(g^2 \log g)$.

Our work:

- Linear bound for exponent[2].
- Constant exponent in the RM-case[3].

Applications:

- Algorithmic questions (deterministic polynomial factorization).
- Program equivalence (Barthe, Jacomme, Kremer, 2020).

---

[2] A., Gaudry, Spaenlehauer. Foundations of Comput. Math., 2019

[3] A. Journal of Complexity, 2020

Main ingredient: multihomogeneous structure

Different strategy: describe I_ℓ with g^2 equations and variables.

Main ingredient: multihomogeneous structure

Different strategy: describe I_ℓ with g^2 equations and variables.

g variables
$O(g^2)$ equations
degree in $O_g(\ell^3)$

$O(g^2)$ variables
$O(g^2)$ equations
degree in $O_g(1)$

Main ingredient: multihomogeneous structure

Different strategy: describe I_ℓ with g^2 equations and variables.

g variables
$O(g^2)$ equations
degree in $O_g(\ell^3)$

$O(g^2)$ variables
$O(g^2)$ equations
degree in $O_g(1)$

Geometric resolution (*Giusti-Lecerf-Salvy'01, Cafure-Matera'06*)

Assume $f_1, \cdots, f_n$ have degrees $\leq d$ and form a reduced regular sequence, and let $\delta = \max_i \deg\langle f_1, \ldots, f_i \rangle$. There is an algorithm computing a geometric resolution in time polynomial in δ, d, n.

Main ingredient: multihomogeneous structure

Different strategy: describe I_ℓ with g^2 equations and variables.

g variables
$O(g^2)$ equations
degree in $O_g(\ell^3)$

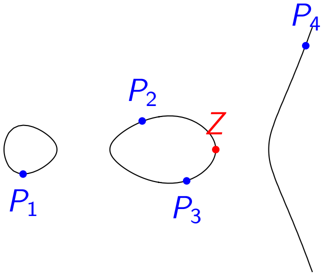
$O(g^2)$ variables
$O(g^2)$ equations
degree in $O_g(1)$

Geometric resolution
(*Giusti-Lecerf-Salvy'01, Cafure-Matera'06*)

Assume $f_1, \cdots, f_n$ have degrees $\leq d$ and form a reduced regular sequence, and let $\delta = \max_i \deg\langle f_1, \ldots, f_i \rangle$. There is an algorithm computing a geometric resolution in time polynomial in δ, d, n.

With $\delta = O_g\left(\ell^{3g}\right)$ bounded by multihomogeneous Bézout bound.
Both $d = O_g(\ell^3)$ and $n = O_g(1)$ are harmless for our bound.

Part II: Riemann-Roch spaces



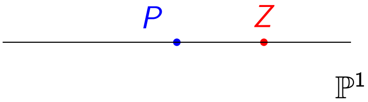
Problem: find all the $\frac{G(X,Y)}{H(X,Y)}$ such that

- Z **must** be a zero of G,
- the P_i **can** be zeroes of H,
- G/H has no other pole.

Applications: arithmetic in Jacobians, AG codes, etc.

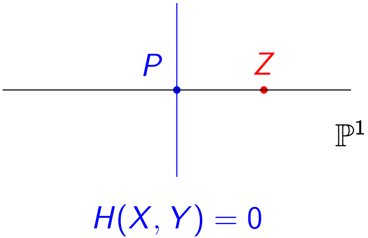
A toy example

Set $\mathcal{C} = \mathbb{P}^1$, $P = [0:1]$, $Z = [1:1]$ and $D = P - Z$.
Previous slide: $\frac{X-1}{X}$ is a solution (one pole in P and one zero in Z).
Riemann-Roch theorem: $\frac{X-1}{X}$ generates the solution space.



A toy example

Set $\mathcal{C} = \mathbb{P}^1$, $P = [0 : 1]$, $Z = [1 : 1]$ and $D = P - Z$.
Previous slide: $\frac{X-1}{X}$ is a solution (one pole in P and one zero in Z).
Riemann-Roch theorem: $\frac{X-1}{X}$ generates the solution space.

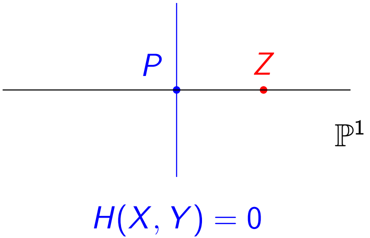


Our strategy

Denominator H passes through P.
This means $H(X, Y) \bmod X = 0$.

A toy example

Set $\mathcal{C} = \mathbb{P}^1$, $P = [0:1]$, $Z = [1:1]$ and $D = P - Z$.
Previous slide: $\frac{X-1}{X}$ is a solution (one pole in P and one zero in Z).
Riemann-Roch theorem: $\frac{X-1}{X}$ generates the solution space.



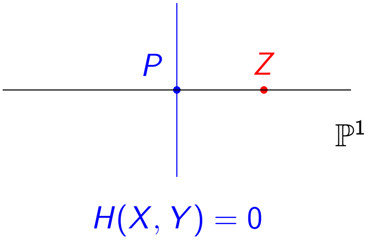
Our strategy

Denominator H passes through P.
This means $H(X,Y) \bmod X = 0$.

Numerators G pass through Z.
It means $G(X,Y) = 0 \bmod (X-1)$.

A toy example

Set $\mathcal{C} = \mathbb{P}^1$, $P = [0 : 1]$, $Z = [1 : 1]$ and $D = P - Z$.
Previous slide: $\frac{X-1}{X}$ is a solution (one pole in P and one zero in Z).
Riemann-Roch theorem: $\frac{X-1}{X}$ generates the solution space.



Our strategy

Denominator H passes through P.
This means $H(X, Y) \bmod X = 0$.

Numerators G pass through Z.
It means $G(X, Y) = 0 \bmod (X - 1)$.

We recover the solution $\frac{X-1}{X}$.

Divisors and Riemann-Roch spaces

Smooth divisor D: finite formal sum $\sum_P m_P P$ of smooth points on $\mathcal{C}$.
Degree of a divisor: $\deg(D) = \sum_P m_P$.

Riemann-Roch space $L(D)$: set of rational functions h such that

- If $m_P < 0$, P **has to be a zero** of h with multiplicity $\geq -m_P$.
- If $m_P > 0$, P **can be a pole** of h with multiplicity $\leq m_P$.

Divisors and Riemann-Roch spaces

Smooth divisor D: finite formal sum $\sum_P m_P P$ of smooth points on $\mathcal{C}$.
Degree of a divisor: $\deg(D) = \sum_P m_P$.

Riemann-Roch space $L(D)$: set of rational functions h such that

- If $m_P < 0$, P **has to be a zero** of h with multiplicity $\geq -m_P$.
- If $m_P > 0$, P **can be a pole** of h with multiplicity $\leq m_P$.

Remember: zeros constrained by D_- and poles allowed by D_+.

Divisors and Riemann-Roch spaces

Smooth divisor D: finite formal sum $\sum_P m_P P$ of smooth points on $\mathcal{C}$.
Degree of a divisor: $\deg(D) = \sum_P m_P$.

Riemann-Roch space $L(D)$: set of rational functions h such that

- If $m_P < 0$, P **has to be a zero** of h with multiplicity $\geq -m_P$.
- If $m_P > 0$, P **can be a pole** of h with multiplicity $\leq m_P$.

Remember: zeros constrained by D_- and poles allowed by D_+.

Our problem:

Given input curve $\mathcal{C}$ and smooth divisor D,
Compute a basis of the vector space $L(D)$.

Geometric vs arithmetic methods

Geometric methods:
Based on Brill-Noether theory.

Arithmetic methods:
Ideals in function fields.

Geometric vs arithmetic methods

Geometric methods:
Based on Brill-Noether theory.

- Goppa, Le Brigand-Risler (80's)
- Huang-Ierardi, Volcheck (90's)
- Khuri-Makdisi (2007)
- Le Gluher-Spaenlehauer (2018)

Arithmetic methods:
Ideals in function fields.

- Coates (1970)
- Davenport (1981)
- Hess's algorithm (2001)

Geometric vs arithmetic methods

Geometric methods:
Based on Brill-Noether theory.

- Goppa, Le Brigand-Risler (80's)
- Huang-Ierardi, Volcheck (90's)
- Khuri-Makdisi (2007)
- Le Gluher-Spaenlehauer (2018)

Arithmetic methods:
Ideals in function fields.

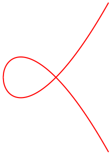
- Coates (1970)
- Davenport (1981)
- Hess's algorithm (2001)

Today: geometric methods

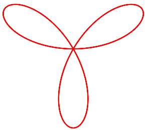
Brill-Noether: belonging conditions for Riemann-Roch spaces.
Conditions $\rightsquigarrow$ linear systems (Le Gluher, Spaenlehauer, 2018).

Geometric algorithms (Brill-Noether theory)

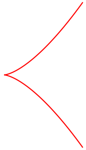
Nodal curve



Ordinary curve



Non-ordinary curve



Our work: conditions $\rightsquigarrow$ belonging to a $K[x]$-module.
Basis of this module through structured linear algebra (Neiger, 2016).

Results: subquadratic algorithms for nodal[4] and ordinary[5] curves.

[4] A., Couvreur, Lecerf. Proceedings of ISSAC 2020

[5] A., Couvreur, Lecerf. Preprint, 2021

A basis of $L(D)$ through Brill-Noether theory

Effective divisors

$D = \sum m_i P_i$ is positive or effective if for any i, $m_i \geq 0$.
Can split $D = D_+ - D_-$ as a difference of two effective divisors.
Denote $D \geq D'$ whenever $D - D'$ is effective.

A basis of $L(D)$ through Brill-Noether theory

Effective divisors

$D = \sum m_i P_i$ is positive or effective if for any i, $m_i \geq 0$.
Can split $D = D_+ - D_-$ as a difference of two effective divisors.
Denote $D \geq D'$ whenever $D - D'$ is effective.

Principal divisor: $(h) = \sum_{P \in \mathcal{C}} \mathrm{ord}_P(h) P$ (zeros−poles with multiplicity)

A basis of $L(D)$ through Brill-Noether theory

Effective divisors

$D = \sum m_i P_i$ is positive or effective if for any i, $m_i \geq 0$.
Can split $D = D_+ - D_-$ as a difference of two effective divisors.
Denote $D \geq D'$ whenever $D - D'$ is effective.

Principal divisor: $(h) = \sum_{P \in \mathcal{C}} \mathrm{ord}_P(h) P$ (zeros−poles with multiplicity)

A description for $L(D)$ (Haché, Le Brigand-Risler)

Non-zero elements of $L(D)$ are of the form G/H where:

- The common denominator H satisfies $(H) \geq D$.
- H must pass through all the singular points of $\mathcal{C}$.
- G is of degree $\deg H$ and $(G) \geq (H) - D$.

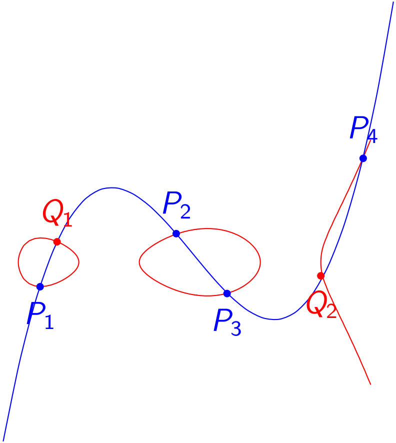
Sketch of the algorithm

Step 1 Find a denominator H.

Step 2 Compute (H).

Step 3 Compute $(H) - D$.

Step 4 Compute numerators.
(Very similar to step 1)



Steps 2 and 3 are a combination of usual techniques.
Let us focus on the interpolation problem of step 1.

Finding a denominator in practice

Conditions on H: passing through singularities and $(H) \geq D_+$.

In primitive form, $(H) \geq D_+ \Leftrightarrow H(X, v_+(X)) = 0 \bmod \chi_+(X)$.
Passing through singularities: similar equation.

Finding a denominator in practice

Conditions on H: passing through singularities and $(H) \geq D_+$.

In primitive form, $(H) \geq D_+ \Leftrightarrow H(X, v_+(X)) = 0 \bmod \chi_+(X)$.
Passing through singularities: similar equation.

Set $d = \deg H$ and write $H = \sum_{i=1}^{d} h_i(X) Y^i$.
Above conditions on H: the h_i's are in a $K[X]$-module of rank $d + 1$.

Finding a denominator in practice

Conditions on H: passing through singularities and $(H) \geq D_+$.

In primitive form, $(H) \geq D_+ \Leftrightarrow H(X, v_+(X)) = 0 \bmod \chi_+(X)$.
Passing through singularities: similar equation.

Set $d = \deg H$ and write $H = \sum_{i=1}^{d} h_i(X) Y^i$.
Above conditions on H: the h_i's are in a $K[X]$-module of rank $d+1$.

Computing a solution basis (Neiger, 2016)

A basis of this $K[X]$-module costs $\widetilde{O}(d^{\omega-1} \deg \chi_+)$ field ops.
Problem: d is unknown, we prove an a priori bound.

Overall complexity exponent: $(\omega + 1)/2$.

Beyond the nodal case

Smooth part: $(H) \geq D_+$ remains $H(X, v(X)) = 0 \bmod \chi(X)$.
Singular part: H passes through singularities with multiplicities.

[5] A., Couvreur, Lecerf. Preprint, 2021

Beyond the nodal case

Smooth part: $(H) \geq D_+$ remains $H(X, v(X)) = 0 \bmod \chi(X)$.
Singular part: H passes through singularities with multiplicities.

Problems:

- How to rephrase Noether's conditions?
  Multiplicities $\rightsquigarrow$ valuation theory, local expansions.
- How to perform the interpolation step?
  Naive extension $\rightsquigarrow$ too many equations, bad complexity.

[5] A., Couvreur, Lecerf. Preprint, 2021

Beyond the nodal case

Smooth part: $(H) \geq D_+$ remains $H(X, v(X)) = 0 \bmod \chi(X)$.
Singular part: H passes through singularities with multiplicities.

Problems:

- How to rephrase Noether's conditions?
  Multiplicities $\rightsquigarrow$ valuation theory, local expansions.
- How to perform the interpolation step?
  Naive extension $\rightsquigarrow$ too many equations, bad complexity.

Complexity bounds

- **Ordinary case**[5] **:** same as nodal (exponent $(\omega + 1)/2$).
- **General case:** ongoing work, target exponent ω first.

[5] A., Couvreur, Lecerf. Preprint, 2021

Prospective

- Point-counting in genus 2 and 3:
  - New algorithms for bivariate resultants
  - Improve Gröbner-based approach (symmetry, further structure)
- Riemann-Roch spaces:
  - Implement fast algorithms through solution bases
  - Handle the non-ordinary case
- Develop a toolbox for efficient AG codes:
  - Algorithms for encoding/decoding
  - New choice of curves based on applications

Thank you for your attention!

Appendix: faster resultants in point-counting

Villard's algorithm for bivariate resultants

Genus	Usual resultants	Villard's algorithm	With $\omega = 2.8$
$g = 2$	$\widetilde{O}(\log^8 q)$	$\widetilde{O}((\log q)^{8-2/\omega})$	$\widetilde{O}((\log q)^{7.3})$
$g = 2$ RM	$\widetilde{O}(\log^5 q)$	$\widetilde{O}((\log q)^{5-1/\omega})^*$	$\widetilde{O}((\log q)^{4.6})^*$
$g = 3$	$\widetilde{O}(\log^{14} q)$	$\widetilde{O}((\log q)^{14-4/\omega})$	$\widetilde{O}((\log q)^{12.6})$
$g = 3$ RM	$\widetilde{O}(\log^6 q)$	$\widetilde{O}((\log q)^{6-4/(3\omega)})$	$\widetilde{O}((\log q)^{5.5})$

Using van der Hoeven and Lecerf's algorithm

Genus	Usual resultants	van der Hoeven - Lecerf
$g = 2$	$\widetilde{O}(\log^8 q)$	$\widetilde{O}((\log q)^6)$
$g = 2$ RM	$\widetilde{O}(\log^5 q)$	$\widetilde{O}((\log q)^4)^*$
$g = 3$	$\widetilde{O}(\log^{14} q)$	$\widetilde{O}((\log q)^{10})$
$g = 3$ RM	$\widetilde{O}(\log^6 q)$	$\widetilde{O}((\log q)^{2+8/3})$