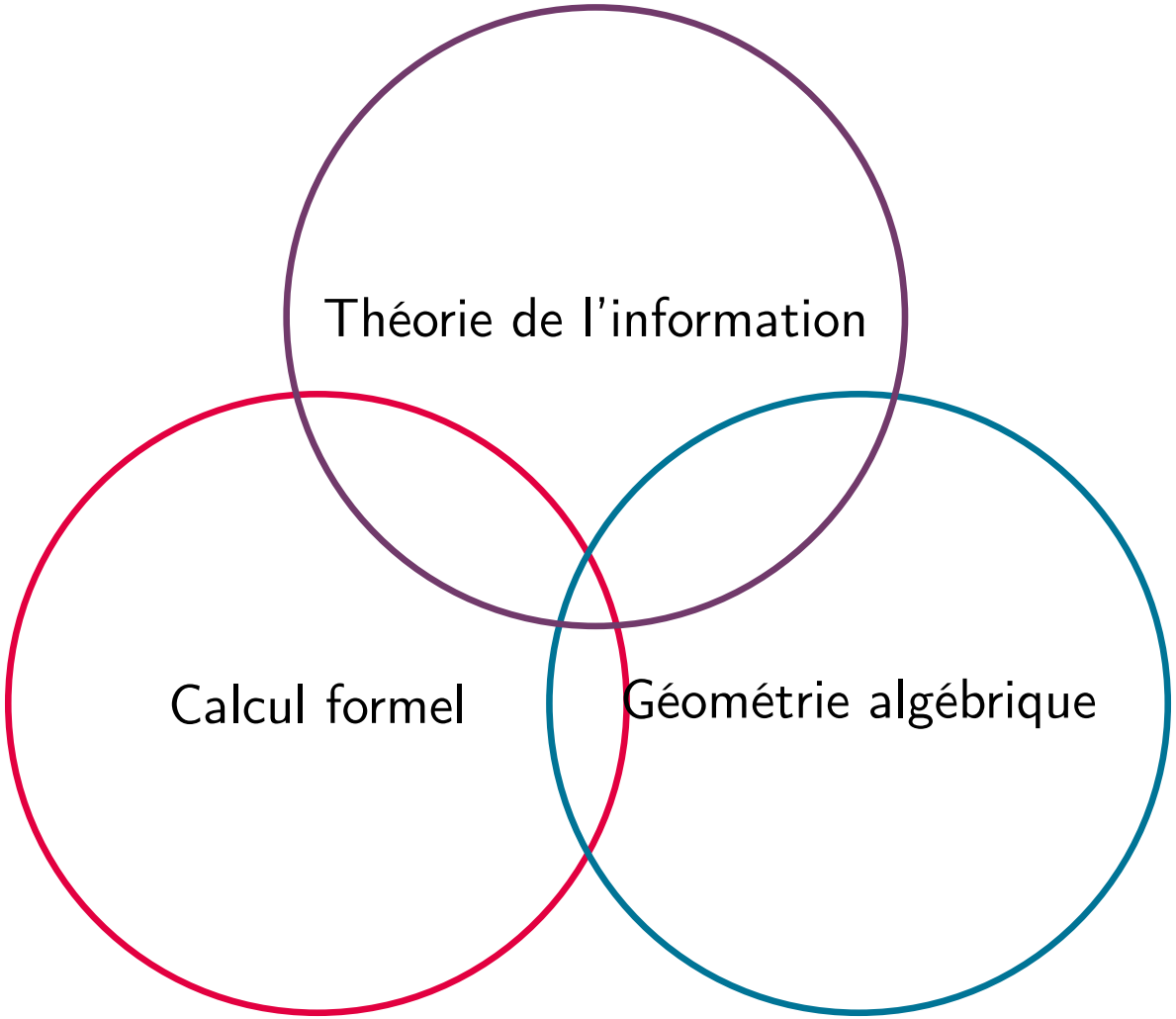
Codes géométriques : mise en œuvre et applications

Elena Berardini

avec S. Abelard (Thales), A. Couvreur (Inria), G. Lecerf (LIX)

Journées de Théorie des Nombres et Cryptographie de Valenciennes
2&3 novembre 2021


Théorie de l'information
Calcul formel
Géométrie algébrique

Sommaire

Qu'est que c'est un code correcteur d'erreur ?

Une méthode pour transmettre et stocker des données.

Caractéristiques : detection et correction des erreurs qui peuvent arriver lors de la transmission / le stockage

Qu'est que c'est un code correcteur d'erreur ?

Une méthode pour transmettre et stocker des données.

Caractéristiques : detection et correction des erreurs qui peuvent arriver lors de la transmission / le stockage

Un $\mathbb{F}_q$-sous espace vectoriel de $\mathbb{F}_q^n$ (codes linéaires).

Trois paramètres :

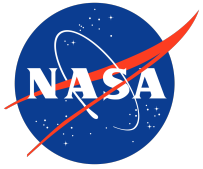
- **n**, la longueur
- **k**, la dimension
- **d**, la distance minimale

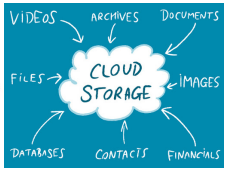
Taux de transmission: k/n
Detecte jusqu'à $d-1$ erreurs
Corrige jusqu'à $\lfloor \frac{d-1}{2} \rfloor$ erreurs

Qu'est que c'est un code correcteur d'erreur ?

Une méthode pour transmettre et stocker des données.

Caractéristiques : detection et correction des erreurs qui peuvent arriver lors de la transmission / le stockage





Un $\mathbb{F}_q$-sous espace vectoriel de $\mathbb{F}_q^n$ (codes linéaires).

Trois paramètres :

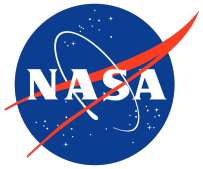
- **n**, la longueur
- **k**, la dimension
- **d**, la distance minimale

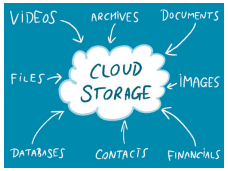
Taux de transmission: k/n
Detecte jusqu'à $d-1$ erreurs
Corrige jusqu'à $\lfloor \frac{d-1}{2} \rfloor$ erreurs

Qu'est que c'est un code correcteur d'erreur ?

Une méthode pour transmettre et stocker des données.

Caractéristiques : detection et correction des erreurs qui peuvent arriver lors de la transmission / le stockage





BUT : encoder le plus grand nombre de données et detecter et corriger le plus grand nombre d'erreurs !

Un $\mathbb{F}_q$-sous espace vectoriel de $\mathbb{F}_q^n$ (codes linéaires).

Trois paramètres :

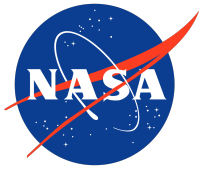
- **n**, la longueur
- **k**, la dimension
- **d**, la distance minimale

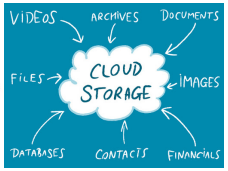
Taux de transmission: k/n
Detecte jusqu'à $d-1$ erreurs
Corrige jusqu'à $\lfloor \frac{d-1}{2} \rfloor$ erreurs

Qu'est que c'est un code correcteur d'erreur ?

Une méthode pour transmettre et stocker des données.

Caractéristiques : detection et correction des erreurs qui peuvent arriver lors de la transmission / le stockage





BUT : encoder le plus grand nombre de données et detecter et corriger le plus grand nombre d'erreurs !

Un $\mathbb{F}_q$-sous espace vectoriel de $\mathbb{F}_q^n$ (codes linéaires).

Trois paramètres :

- **n**, la longueur
- **k**, la dimension
- **d**, la distance minimale

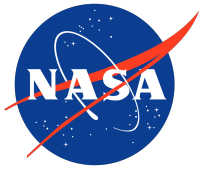
Taux de transmission: k/n
Detecte jusqu'à $d-1$ erreurs
Corrige jusqu'à $\lfloor \frac{d-1}{2} \rfloor$ erreurs

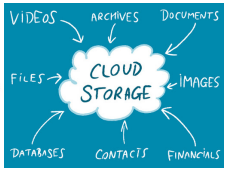
BUT : avoir **k** et **d** les plus grands possible !

Qu'est que c'est un code correcteur d'erreur ?

Une méthode pour transmettre et stocker des données.

Caractéristiques : detection et correction des erreurs qui peuvent arriver lors de la transmission / le stockage





BUT : encoder le plus grand nombre de données et detecter et corriger le plus grand nombre d'erreurs !

Un $\mathbb{F}_q$-sous espace vectoriel de $\mathbb{F}_q^n$ (codes linéaires).

Trois paramètres :

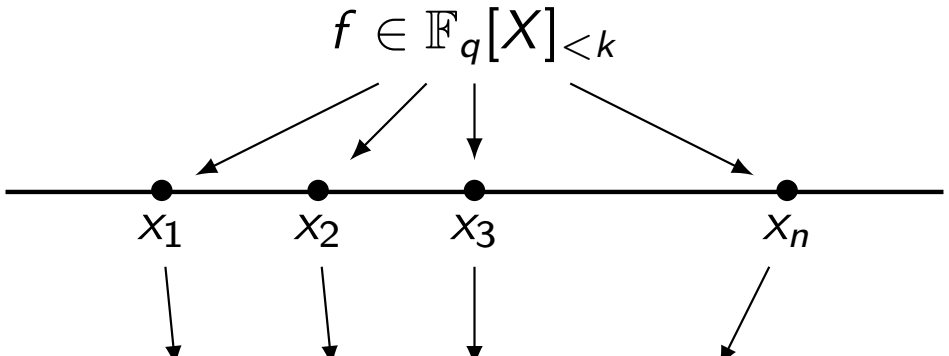
- **n**, la longueur
- **k**, la dimension
- **d**, la distance minimale

Taux de transmission: k/n
Detecte jusqu'à $d-1$ erreurs
Corrige jusqu'à $\lfloor \frac{d-1}{2} \rfloor$ erreurs

BUT : avoir **k** et **d** les plus grands possible !

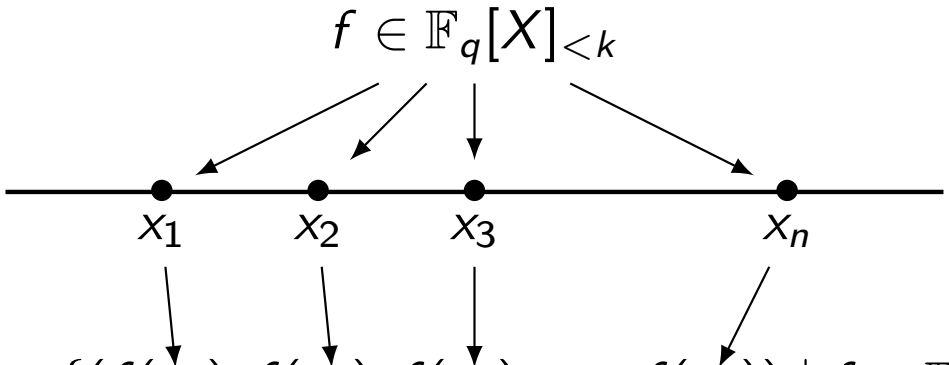
Borne de Singleton : $k + d \leqslant n + 1$
$\rightsquigarrow$ compromis entre redondance et capacité de correction

Codes d'évaluation : des codes de Reed–Solomon...



$$\mathrm{RS}_k(\mathbf{x}) := \{(f(x_1), f(x_2), f(x_3), \ldots, f(x_n)) \mid f \in \mathbb{F}_q[X]_{<k}\}$$

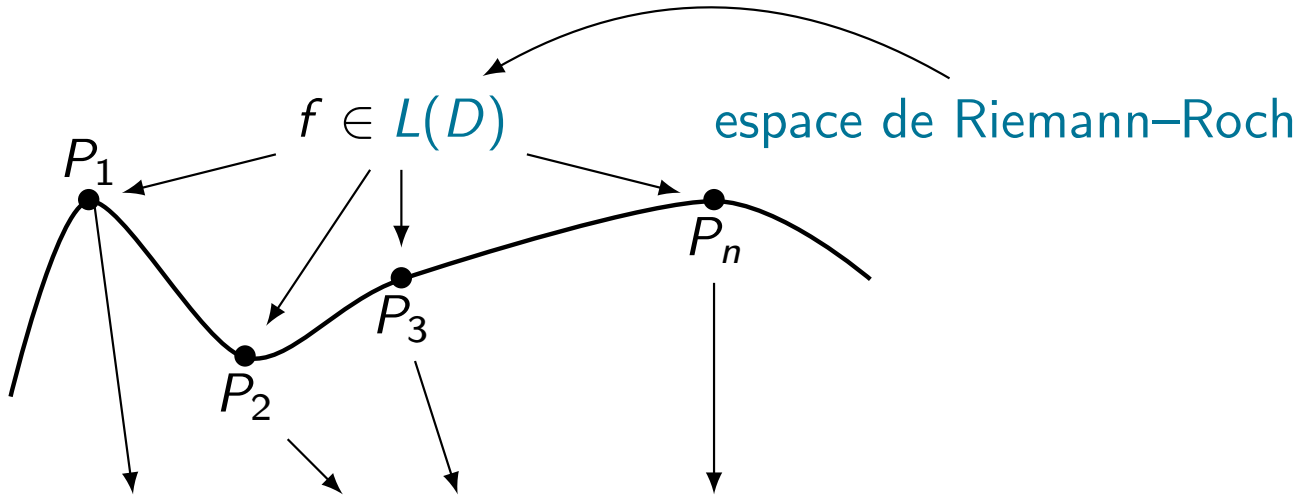
Codes d'évaluation : des codes de Reed–Solomon...



$$\mathrm{RS}_k(\mathbf{x}) := \{(f(x_1), f(x_2), f(x_3), \ldots, f(x_n)) \mid f \in \mathbb{F}_q[X]_{<k}\}$$

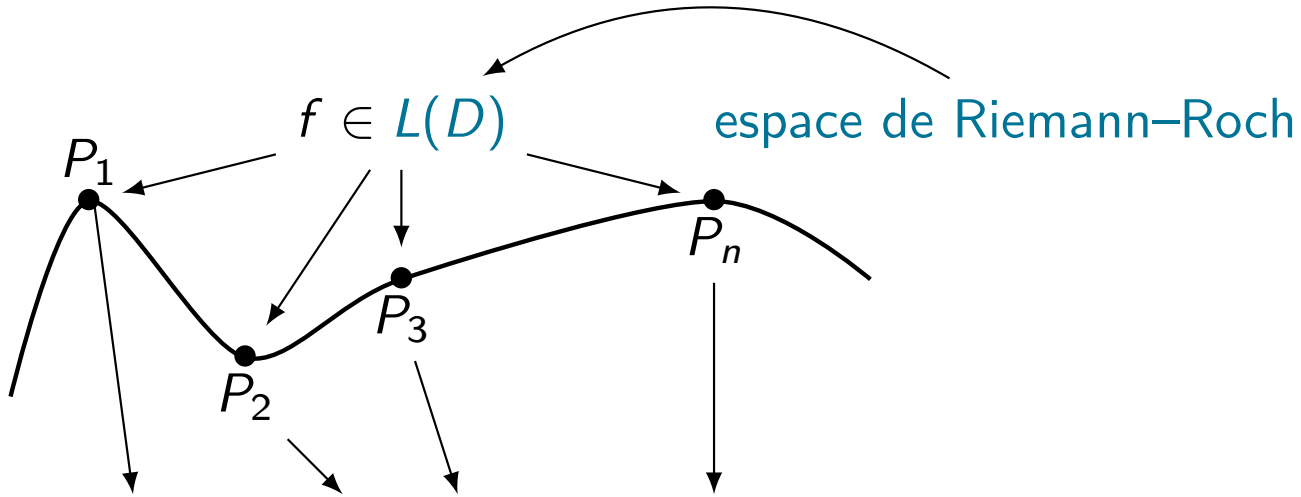
- ✓ Paramètres optimaux : $k + d = n + 1$ (codes MDS)
- ✓ Algorithme de décodage efficace (Berlekamp, 1968)
- ✓ Operations sur les données
- ⚠ Incovenient : $n \leqslant q$

...aux codes géométriques



$$\mathcal{C}((P_i)_i, D) := \{(f(P_1), f(P_2), f(P_3), \ldots, f(P_n)) \mid f \in L(D)\}$$

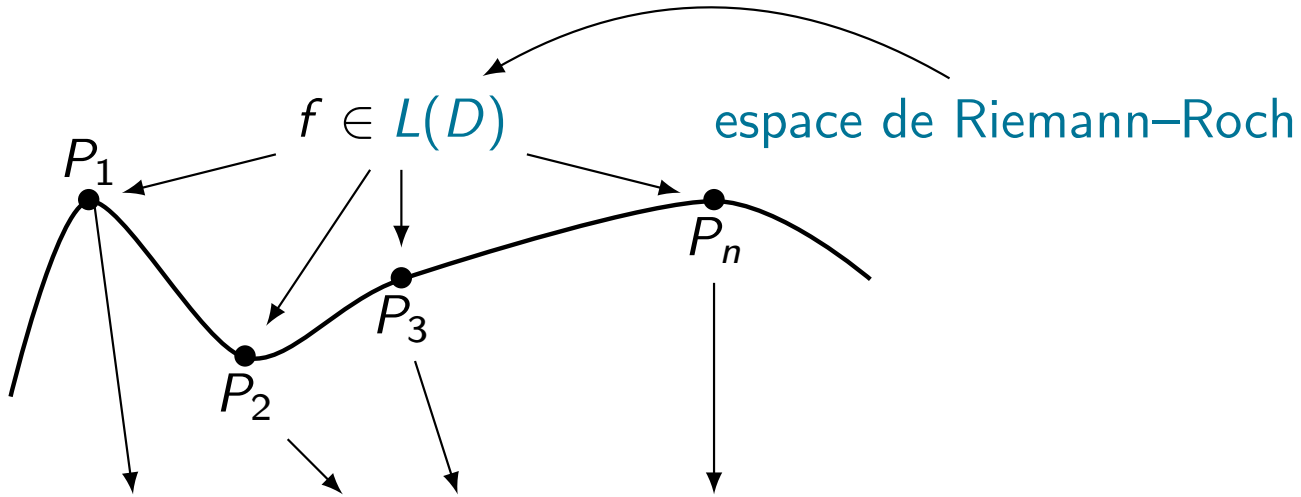
...aux codes géométriques



$$\mathcal{C}((P_i)_i, D) := \{(f(P_1), f(P_2), f(P_3), \ldots, f(P_n)) \mid f \in L(D)\}$$

Longueur : $|\#C(\mathbb{F}_q) - (q+1)| \leq g\lfloor 2\sqrt{q} \rfloor$

...aux codes géométriques



$$\mathcal{C}((P_i)_i, D) := \{(f(P_1), f(P_2), f(P_3), \ldots, f(P_n)) \mid f \in L(D)\}$$

Longeur : $|\#C(\mathbb{F}_q) - (q+1)| \leq g\lfloor 2\sqrt{q}\rfloor$

Proposition

Les paramètres $[n, k, d]$ des codes géométriques satisfont

$$n + 1 - g \leq k + d \leq n + 1.$$

$\rightsquigarrow$ les codes AG sont à distance g de l'optimalité

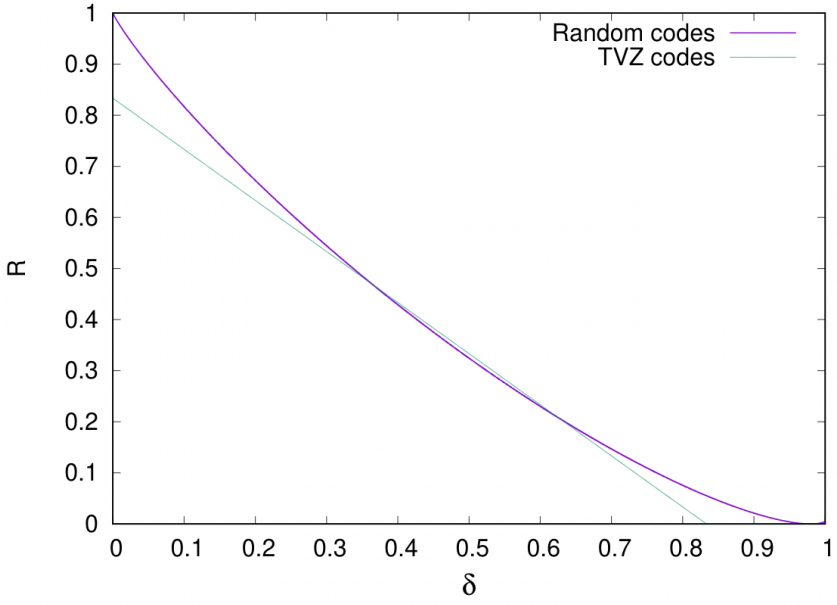
Bref histoire des codes géométriques

1981: Goppa introduit les codes AG sur les courbes algébriques

Bref histoire des codes géométriques

1981: Goppa introduit les codes AG sur les courbes algébriques

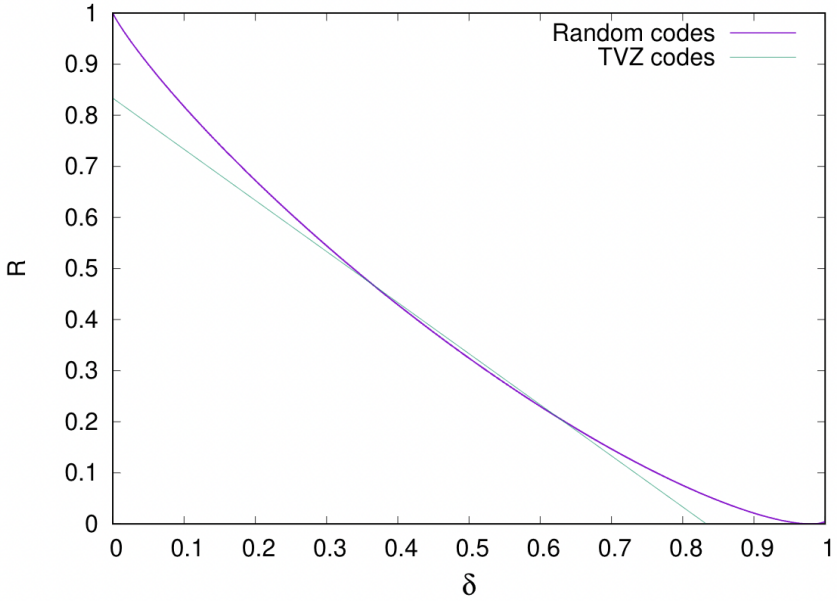
1982: Tsfasman, Vlăduț et Zink utilisent les codes AG pour dépasser la borne de Gilbert–Varshamov



Bref histoire des codes géométriques

1981: Goppa introduit les codes AG sur les courbes algébriques

1982: Tsfasman, Vlăduț et Zink utilisent les codes AG pour dépasser la borne de Gilbert–Varshamov



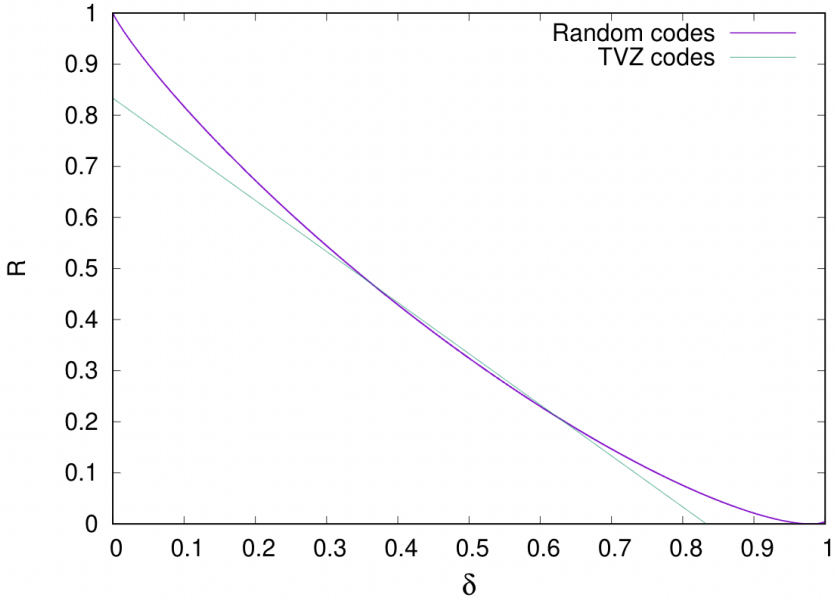
XXc: des different familles de courbes sont étudiées afin d'obtenir des codes optimaux ou quasi–optimaux

$\hookrightarrow$ seulement les courbes dont les espaces de Riemann–Roch sont déjà connus sont utilisées

Bref histoire des codes géométriques

1981: Goppa introduit les codes AG sur les courbes algébriques

1982: Tsfasman, Vlăduț et Zink utilisent les codes AG pour dépasser la borne de Gilbert–Varshamov



XXc: des different familles de courbes sont étudiées afin d'obtenir des codes optimaux ou quasi–optimaux

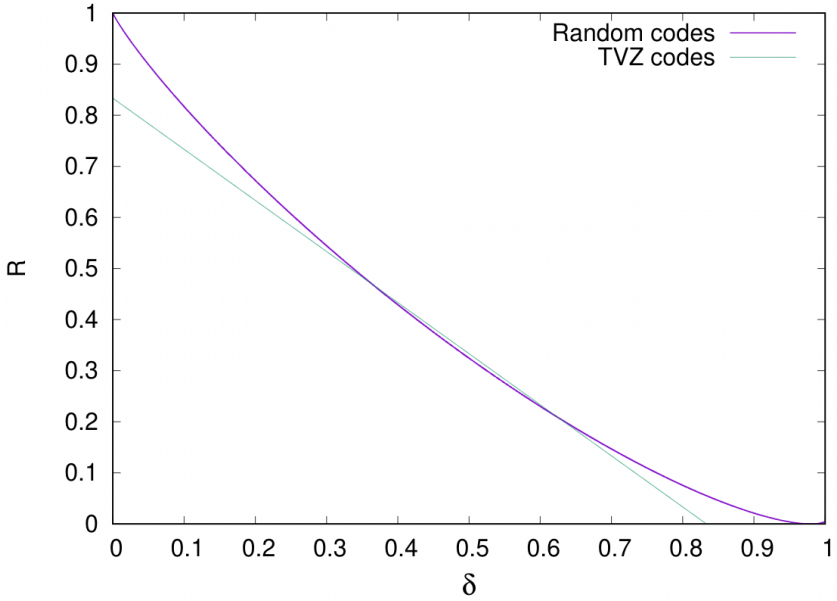
↪ seulement les courbes dont les espaces de Riemann–Roch sont déjà connus sont utilisées

XXIc: les codes AG sont utilisés dans des nouvelles applications en théorie de l'information

Bref histoire des codes géométriques

1981: Goppa introduit les codes AG sur les courbes algébriques

1982: Tsfasman, Vlăduț et Zink utilisent les codes AG pour dépasser la borne de Gilbert–Varshamov



XXc: des different familles de courbes sont étudiées afin d'obtenir des codes optimaux ou quasi–optimaux

↪ seulement les courbes dont les espaces de Riemann–Roch sont déjà connus sont utilisées

XXIc: les codes AG sont utilisés dans des nouvelles applications en théorie de l'information ...allons voir comment !

Cryptosystème McEliece (1978)

Cryptosystème McEliece (1978)

- G, matrice d'un code $[n, k, 2t+1]$
- $\mathcal{A}$, algorithme de décodage
- S, a $k \times k$ matrice
- P, a $n \times n$ matrice

Calcule $\bar{G} = SGP$

PubKey $= (\bar{G}, t)$, SecKey$= (G, P, S, \mathcal{A})$

Cryptosystème McEliece (1978)

- G, matrice d'un code $[n, k, 2t+1]$
- $\mathcal{A}$, algorithme de décodage
- S, a $k \times k$ matrice
- P, a $n \times n$ matrice

Calcule $\bar{G} = SGP$

PubKey $= (\bar{G}, t)$, SecKey$= (G, P, S, \mathcal{A})$

- Divise son message en vecteurs m_i de longueur k
- Construit un vecteur e de longueur n et poids t

Cryptosystème McEliece (1978)

- G, matrice d'un code $[n, k, 2t+1]$
- $\mathcal{A}$, algorithme de décodage
- S, a $k \times k$ matrice
- P, a $n \times n$ matrice

Calcule $\bar{G} = SGP$

PubKey $= (\bar{G}, t)$, SecKey$= (G, P, S, \mathcal{A})$

- Divise son message en vecteurs m_i de longueur k
- Construit un vecteur e de longueur n et poids t

Calcule

$$y_i = m_i \bar{G} + e$$

Cryptosystème McEliece (1978)

- G, matrice d'un code $[n, k, 2t+1]$
- $\mathcal{A}$, algorithme de décodage
- S, a $k \times k$ matrice
- P, a $n \times n$ matrice

Calcule $\bar{G} = SGP$

PubKey $= (\bar{G}, t)$, SecKey$= (G, P, S, \mathcal{A})$

- Divise son message en vecteurs m_i de longueur k
- Construit un vecteur e de longueur n et poids t

Calcule

$$y_i = m_i \bar{G} + e$$

Reçoit y_i

Cryptosystème McEliece (1978)

- G, matrice d'un code $[n, k, 2t+1]$
- $\mathcal{A}$, algorithme de décodage
- S, a $k \times k$ matrice
- P, a $n \times n$ matrice

Calcule $\bar{G} = SGP$

PubKey $= (\bar{G}, t)$, SecKey$= (G, P, S, \mathcal{A})$

- Divise son message en vecteurs m_i de longueur k
- Construit un vecteur e de longueur n et poids t

Calcule
$y_i = m_i\bar{G} + e$

Reçoit y_i

Calcule $y_i P^{-1} = (m_i\bar{G} + e)P^{-1}$
$= m_i SG + eP^{-1} = m_i SG + e'$
Utilise $\mathcal{A}$ pour retrouver $m_i SG$

$$m_i = m_i SG \times G^{-1}S^{-1}$$

Cryptosystème McEliece (1978)

- G, matrice d'un code $[n, k, 2t+1]$
- $\mathcal{A}$, algorithme de décodage
- S, a $k \times k$ matrice
- P, a $n \times n$ matrice

Calcule $\bar{G} = SGP$

PubKey $= (\bar{G}, t)$, SecKey$= (G, P, S, \mathcal{A})$

- Divise son message en vecteurs m_i de longueur k
- Construit un vecteur e de longueur n et poids t

Calcule
$y_i = m_i\bar{G} + e$

Reçoit y_i

Calcule $y_i P^{-1} = (m_i\bar{G} + e)P^{-1}$
$= m_i SG + eP^{-1} = m_i SG + e'$
Utilise $\mathcal{A}$ pour retrouver $m_i SG$

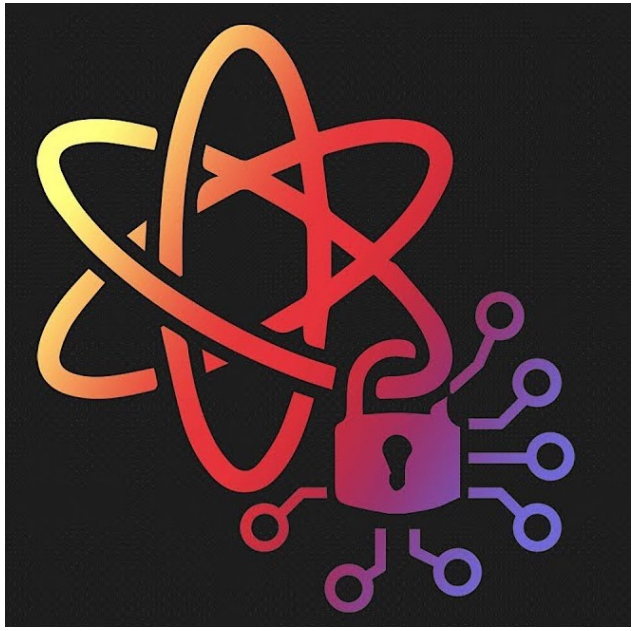
$$m_i = m_i SG \times G^{-1}S^{-1}$$

Le cryptosystème McEliece pour la crypto post-quantique

- la sécurité se base sur la
  - difficulté calculatoire du décodage d'un code aléatoire
  - difficulté calculatoire de distinguer un code structuré d'un code aléatoire

✓ Post-quantum

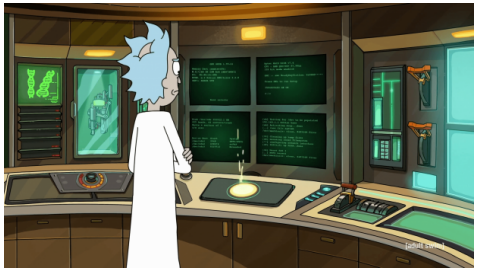
⚠ Nécessite des clés de grande taille



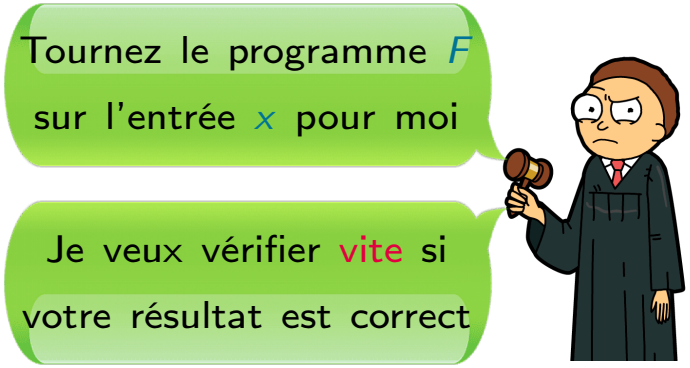
⚠ Les codes AG classiques ne sont pas sûrs pour ce cryptosystème

✓ Les sous-codes de codes AG sur un sous-corps sont prometteurs !

Calcul Vérifiable

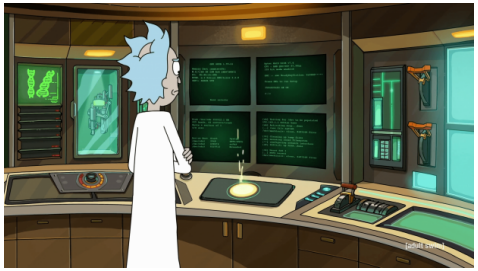


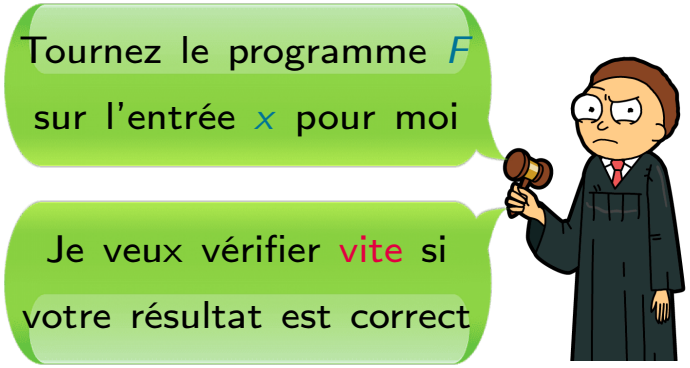
Prouver Puissant
(e.g. un serveur)



Faible Vérifieur
(e.g. un client)

Calcul Vérifiable





Prouver Puissant
(e.g. un serveur)

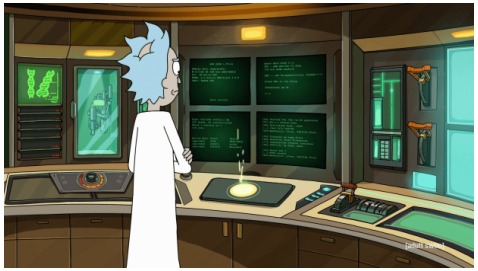
retourne le résultat y et
une preuve de correction π

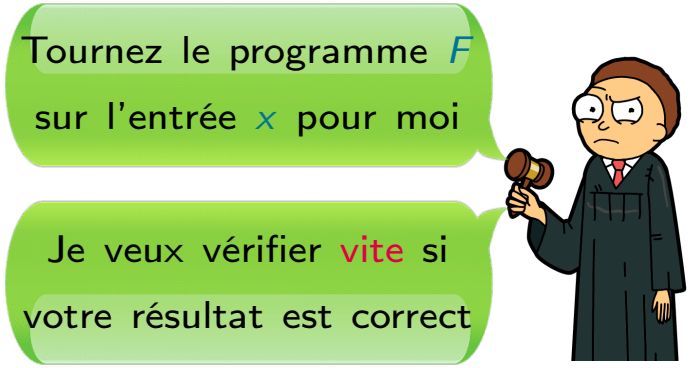
$\xrightarrow{y,\pi}$

Faible Vérifieur
(e.g. un client)

vérifie la validité de π
pour l'énoncé $y = F(x)$

Calcul Vérifiable





Prouver Puissant
(e.g. un serveur)

retourne le résultat y et
une preuve de correction π

$\xrightarrow{y,\pi}$

Faible Vérifieur
(e.g. un client)

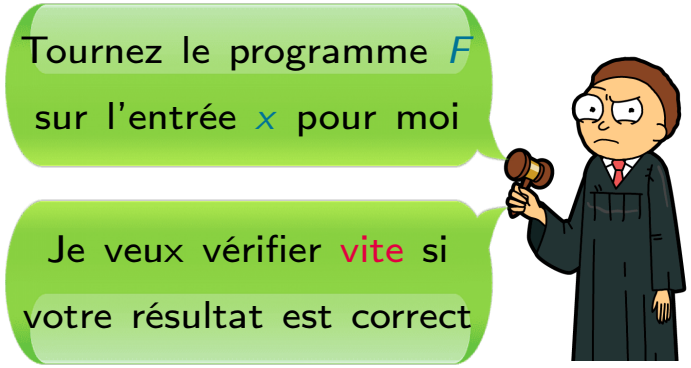
vérifie la validité de π
pour l'énoncé $y = F(x)$

Proximité à un code C

Le Prouver renvoie un mot

- $c \in C$, si l'énoncé $y = F(x)$ est vrai,
- $\tilde{c}$ très loin de C, autrement.

Calcul Vérifiable



Prouver Puissant
(e.g. un serveur)

retourne le résultat y et
une preuve de correction π

$\xrightarrow{y,\pi}$

Faible Vérifieur
(e.g. un client)

vérifie la validité de π
pour l'énoncé $y = F(x)$

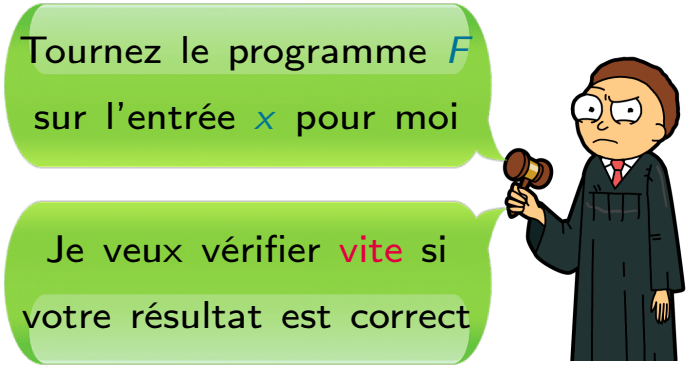
Proximité à un code C

Le Prouver renvoie un mot

- $c \in C$, si l'énoncé $y = F(x)$ est vrai,
- $\tilde{c}$ très loin de C, autrement.

Applications : cryptomonnaies, blockchain...

Calcul Vérifiable



Prouver Puissant
(e.g. un serveur)

retourne le résultat y et une preuve de correction π

$\xrightarrow{y,\pi}$

Faible Vérifieur
(e.g. un client)

vérifie la validité de π pour l'énoncé $y = F(x)$

Proximité à un code C

Le Prouver renvoie un mot

- $c \in C$, si l'énoncé $y = F(x)$ est vrai,
- $\tilde{c}$ très loin de C, autrement.

Applications : cryptomonnaies, blockchain...

Quels codes peuvent être utilisés ? Les codes AG semblent être une bonne option[1]

[1] S. Bordage et J. Nardi, preprint, 2020

Espaces de Riemann–Roch : les codes AG et au–delà

Construction explicites de codes AG pour

- Cryptosystème de McEliece
- Calcul Vérifiable
- Autres applications (partage de secret, stockage distribué...)

$\rightsquigarrow$ besoin de calculer les espaces de Riemann–Roch de courbes

Espaces de Riemann–Roch : les codes AG et au–delà

Construction explicites de codes AG pour

- Cryptosystème de McEliece
- Calcul Vérifiable
- Autres applications (partage de secret, stockage distribué...)

$\rightsquigarrow$ besoin de calculer les espaces de Riemann–Roch de courbes

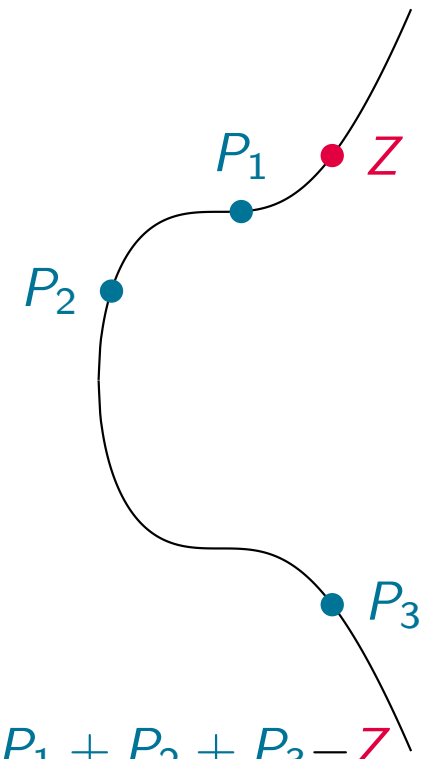
Cela est utile aussi pour...

- Operations arithmétiques sur les Jacobiennes de courbes[2]
- Integration symbolique[3]

[2] K. Khuri-Makdisi, Mathematics of Computations, 2007
[3] J.H. Davenport, Intern. Symp. on Symbolic et Algebraic Manipulation, 1979

Espaces de Riemann–Roch

Diviseurs sur une courbe $\mathcal{C}$: $D = \sum_{P \in \mathcal{C}} n_P P$



D$=P_1 + P_2 + P_3 - Z$

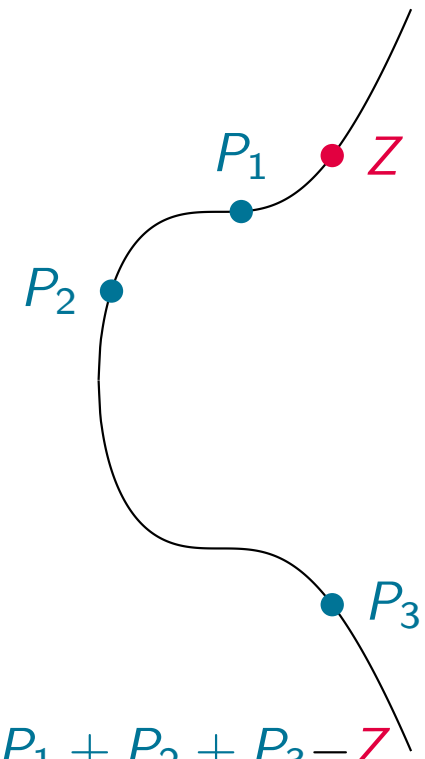
L'**espace de Riemann–Roch** $L(D)$ est l'espace de toutes les fonctions de la forme $\frac{G}{H} \in \mathbb{K}(\mathcal{C})$ telles que :

- si $n_P < 0$ alors P doit être un zéro de G (de multiplicité $\geqslant -n_P$)
- si $n_P > 0$ alors P peut être un zéro of H (de multiplicité $\leqslant n_P$)
- G/H n'a pas d'autres pôles en dehors des points P avec $n_P > 0$

Ici : Z doit être un zéro de G, les P_i peuvent être des zéros de H

Espaces de Riemann–Roch

Diviseurs sur une courbe $\mathcal{C}$: $D = \sum_{P \in \mathcal{C}} n_P P$



D=$P_1 + P_2 + P_3 - Z$

L'**espace de Riemann–Roch** $L(D)$ est l'espace de toutes les fonctions de la forme $\frac{G}{H} \in \mathbb{K}(\mathcal{C})$ telles que :

- si $n_P < 0$ alors P doit être un zéro de G (de multiplicité $\geqslant -n_P$)
- si $n_P > 0$ alors P peut être un zéro of H (de multiplicité $\leqslant n_P$)
- G/H n'a pas d'autres pôles en dehors des points P avec $n_P > 0$

Ici : Z doit être un zéro de G, les P_i peuvent être des zéros de H

Théorème de Riemann–Roch $\rightsquigarrow$ dimension de $L(D) = \deg D + 1 - g$

où le degré d'un diviseur est $\deg D = \sum_P n_P$

Exemple jouet

Soit $\mathcal{C} = \mathbb{P}^1$, $P = [0:1]$ et $Q = [1:1]$. Soit $D = P - Q$, alors

$$f \in L(D) \iff \begin{cases} \text{f a un zéro d'ordre au moins 1 en } Q \\ \text{f peut avoir un pôle d'ordre au plus 1 en } P \\ \text{f n'a pas d'autres pôles en dehors de } P \end{cases}$$

Exemple jouet

Soit $\mathcal{C} = \mathbb{P}^1$, $P = [0:1]$ et $Q = [1:1]$. Soit $D = P - Q$, alors

$$f \in L(D) \iff \begin{cases} \text{f a un zéro d'ordre au moins 1 en } Q \\ \text{f peut avoir un pôle d'ordre au plus 1 en } P \\ \text{f n'a pas d'autres pôles en dehors de } P \end{cases}$$

$$f = \frac{X-1}{X} \text{ est une solution}$$

$$g = 0, \deg D = 0 \xrightarrow[\text{Riemann–Roch}]{\text{Théorème de}} \dim L(D) = \deg D + 1 - g = 1$$

$$\rightarrow f \text{ engendre l'espace des solutions}$$

Exemple jouet

Soit $\mathcal{C} = \mathbb{P}^1$, $P = [0:1]$ et $Q = [1:1]$. Soit $D = P - Q$, alors

$$f \in L(D) \iff \begin{cases} \text{f a un zéro d'ordre au moins 1 en } Q \\ \text{f peut avoir un pôle d'ordre au plus 1 en } P \\ \text{f n'a pas d'autres pôles en dehors de } P \end{cases}$$

$$f = \frac{X-1}{X} \text{ est une solution}$$

$$g = 0, \deg D = 0 \xrightarrow[\text{Riemann–Roch}]{\text{Théorème de}} \dim L(D) = \deg D + 1 - g = 1$$

$$\rightarrow f \text{ engendre l'espace des solutions}$$

⚠on n'a pas une méthode explicite pour calculer une base de $L(D)$
Comment résoudre le problème en général?

Problème de Riemann–Roch : état de l'art

Méthode géométrique :
(Théorie de Brill–Noether ∼1874)

- Goppa, Le Brigand–Risler (80's)
- Huang–Ierardi (90's)
- Khuri-Makdisi (2007)
- Le Gluher–Spaenlehauer (2018)
- Abelard–Couvreur–Lecerf (2020)

Méthode arithmétique :
(Idéaux dans de corps de fonctions)

- Hensel–Landberg (1902)
- Coates (1970)
- Davenport (1981)
- Hess (2001)

Problème de Riemann–Roch : état de l'art

Méthode géométrique :
(Théorie de Brill–Noether ∼1874)
- Goppa, Le Brigand–Risler (80's)
- Huang–Ierardi (90's)
- Khuri-Makdisi (2007)
- Le Gluher–Spaenlehauer (2018)
- Abelard–Couvreur–Lecerf (2020)

Méthode arithmétique :
(Idéaux dans de corps de fonctions)
- Hensel–Landberg (1902)
- Coates (1970)
- Davenport (1981)
- Hess (2001)

Courbes ordinaires/nodales : Algorithme Las Vegas qui calcule $L(D)$ en $\tilde{\mathcal{O}}((\delta^2 + \deg D)^{\frac{\omega+1}{2}})$ operations[4]

Courbes non–ordinaire : ⚠aucun exposant de complexité explicite

[4] $2 \leqslant \omega \leqslant 3$ est un exposant faisable pour l'algèbre linéaire ($\omega = 2.373$)

Brill–Noether

Méthode Brill–Noether $\rightsquigarrow$ conditions NS sur H et G t.q. $G/H \in L(D)$

Brill–Noether

Méthode Brill–Noether $\rightsquigarrow$ conditions NS sur H et G t.q. $G/H \in L(D)$

Notations :

- $(H) = \sum_{P \in \mathcal{C}} \mathrm{ord}_P(H)P$ (zéros de H avec multiplicité)
- $D \geqslant D' \rightsquigarrow D - D' = \sum n_P P$ avec $n_P \geqslant 0$ pour tout P

Brill–Noether

Méthode Brill–Noether $\rightsquigarrow$ conditions NS sur H et G t.q. $G/H \in L(D)$

Notations :

- $(H) = \sum_{P \in \mathcal{C}} \mathrm{ord}_P(H)P$ (zéros de H avec multiplicité)
- $D \geqslant D' \rightsquigarrow D - D' = \sum n_P P$ avec $n_P \geqslant 0$ pour tout P

Description de $L(D)$ pour $\mathcal{C} : F(X, Y, Z) = 0$ courbe plane projective.

Les éléments non–nuls sont de la forme $\frac{G_i}{H}$ où

- *H satisfait $(H) \geqslant D$*
- *H passe à travers tout point singulier de $\mathcal{C}$ avec multiplicité ad hoc*
- $\deg G_i = \deg H$, *G_i copremier avec F et $(G_i) \geqslant (H) - D$*

Brill–Noether

Méthode Brill–Noether $\rightsquigarrow$ conditions NS sur H et G t.q. $G/H \in L(D)$

Notations :

- $(H) = \sum_{P \in \mathcal{C}} \mathrm{ord}_P(H)P$ (zéros de H avec multiplicité)
- $D \geqslant D' \rightsquigarrow D - D' = \sum n_P P$ avec $n_P \geqslant 0$ pour tout P

Description de $L(D)$ pour $\mathcal{C} : F(X, Y, Z) = 0$ courbe plane projective.

Les éléments non–nuls sont de la forme $\frac{G_i}{H}$ où

- *H satisfait $(H) \geqslant D$*
- *H passe à travers tout point singulier de $\mathcal{C}$ avec multiplicité ad hoc*
- $\deg G_i = \deg H$, *G_i copremier avec F et $(G_i) \geqslant (H) - D$*

Comment gérer les points singuliers ?

Brill–Noether

Méthode Brill–Noether $\rightsquigarrow$ conditions NS sur H et G t.q. $G/H \in L(D)$

Notations :

- $(H) = \sum_{P \in \mathcal{C}} \mathrm{ord}_P(H)P$ (zéros de H avec multiplicité)
- $D \geqslant D' \rightsquigarrow D - D' = \sum n_P P$ avec $n_P \geqslant 0$ pour tout P

Description de $L(D)$ pour $\mathcal{C} : F(X, Y, Z) = 0$ courbe plane projective.

Les éléments non–nuls sont de la forme $\frac{G_i}{H}$ où

- *H satisfait $(H) \geqslant D$*
- *H satisfait $(H) \geqslant \mathcal{A}$ (on dira que "H est adjoint à la courbe")*
- $\deg G_i = \deg H$, *G_i copremier avec F et $(G_i) \geqslant (H) - D$*

Comment gérer les points singuliers ?

$\rightsquigarrow$ le diviseur d'adjonction $\mathcal{A}$ "contient" les points singuliers de $\mathcal{C}$ avec leurs multiplicités

Brill–Noether

Méthode Brill–Noether $\rightsquigarrow$ conditions NS sur H et G t.q. $G/H \in L(D)$

Notations :

- $(H) = \sum_{P \in \mathcal{C}} \mathrm{ord}_P(H)P$ (zéros de H avec multiplicité)
- $D \geqslant D' \rightsquigarrow D - D' = \sum n_P P$ avec $n_P \geqslant 0$ pour tout P

Description de $L(D)$ pour $\mathcal{C} : F(X, Y, Z) = 0$ courbe plane projective.

Les éléments non–nuls sont de la forme $\frac{G_i}{H}$ où

- *H satisfait $(H) \geqslant D$*
- *H satisfait$(H) \geqslant \mathcal{A}$*
- $\deg G_i = \deg H$, *G_i copremier avec F et $(G_i) \geqslant (H) - D$*

Comment gérer les points singuliers ?

$\rightsquigarrow$ le diviseur d'adjonction $\mathcal{A}$ "contient" les points singuliers de $\mathcal{C}$ avec leurs multiplicités

Comment gérer les diviseurs?

Brill–Noether

Méthode Brill–Noether $\rightsquigarrow$ conditions NS sur H et G t.q. $G/H \in L(D)$

Notations :

- $(H) = \sum_{P \in \mathcal{C}} \mathrm{ord}_P(H)P$ (zéros de H avec multiplicité)
- $D \geqslant D' \rightsquigarrow D - D' = \sum n_P P$ avec $n_P \geqslant 0$ pour tout P

Description de $L(D)$ pour $\mathcal{C} : F(X, Y, Z) = 0$ courbe plane projective.

Les éléments non–nuls sont de la forme $\frac{G_i}{H}$ où

- *H satisfait $(H) \geqslant D$*
- *H satisfait$(H) \geqslant \mathcal{A}$*
- $\deg G_i = \deg H$, *G_i copremier avec F et $(G_i) \geqslant (H) - D$*

Comment gérer les points singuliers ?

$\rightsquigarrow$ le diviseur d'adjonction $\mathcal{A}$ "contient" les points singuliers de $\mathcal{C}$ avec leurs multiplicités

Comment gérer les diviseurs?

expansions en séries de representations multi-set $((P_i)_i, m_i)$	$\rightsquigarrow$	opérations sur les diviseurs ont coût négligeable

Sketch de l'algorithm

Input

$\mathcal{C} : F(X, Y, Z) = 0$ *une courbe plane projective,* D *un diviseur lisse.*

Étape 1 : Calcule le diviseur d'adjonction $\mathcal{A}$

Étape 2 : Calcule le dénominateur commun H

Étape 3 : Calcule $(H) - D$

Étape 4 : Calcule des numérateurs G_i (proche de l'étape 2)

Output

Une base de l'espace de Riemann–Roch $L(D)$ *en termes de* H *et des* G_i.

Sketch de l'algorithm

Input

$\mathcal{C} : F(X, Y, Z) = 0$ *une courbe plane projective,* D *un diviseur lisse.*

Étape 1 : Calcule le diviseur d'adjonction $\mathcal{A}$

Étape 2 : Calcule le dénominateur commun H

Étape 3 : Calcule $(H) - D$ ✓ $\leftarrow \tilde{O}((\delta^2 + \deg D)^2)$

Étape 4 : Calcule des numérateurs G_i (proche de l'étape 2)

Output

Une base de l'espace de Riemann–Roch $L(D)$ *en termes de* H *et des* G_i.

Sketch de l'algorithm

Input

$\mathcal{C} : F(X, Y, Z) = 0$ *une courbe plane projective,* D *un diviseur lisse.*

Étape 1 : Calcul le diviseur d'adjonction $\mathcal{A}$

Étape 2 : Calcule le dénominateur commun H

Étape 3 : Calcule $(H) - D$ ✓ $\leftarrow \tilde{O}((\delta^2 + \deg D)^2)$

Étape 4 : Calcule des numérateurs G_i (proche de l'étape 2)

Output

Une base de l'espace de Riemann–Roch $L(D)$ *en termes de* H *et des* G_i.

La condition d'adjonction via les séries de Puiseux

Soit $F \in \mathbb{K}[x, y]$ absolument irréductible, monic en y et de degré d en y. Les racines de $F \in \mathbb{K}((x))[y]$ en $\cup_{e \geqslant 1} \overline{\mathbb{K}}((x^{1/e}))$ sont ses series de Puiseux $\varphi_0, \ldots, \varphi_{d-1}$, et F s'écrit

$$F = \prod_{i=1}^{d-1} (y - \varphi_i) = \prod_{i=1}^{d-1} (y - \sum_{j=n}^{\infty} \beta_{i,j} x^{j/e_i}).$$

Exemple : $F = y^2 - x^3 \rightsquigarrow F = (y - x^{3/2})(y + x^{3/2})$

La condition d'adjonction via les séries de Puiseux

Soit $F \in \mathbb{K}[x, y]$ absolument irréductible, monic en y et de degré d en y.
Les racines de $F \in \mathbb{K}((x))[y]$ en $\cup_{e \geqslant 1} \overline{\mathbb{K}}((x^{1/e}))$ sont ses series de Puiseux $\varphi_0, \ldots, \varphi_{d-1}$, et F s'écrit

$$F = \prod_{i=1}^{d-1} (y - \varphi_i) = \prod_{i=1}^{d-1} (y - \sum_{j=n}^{\infty} \beta_{i,j} x^{j/e_i}).$$

Exemple : $F = y^2 - x^3 \rightsquigarrow F = (y - x^{3/2})(y + x^{3/2})$

On fixe une φ de degré e. Soit ζ une racine primitive e-ème de l'unité.
Pour $0 \leqslant k < e$ on peut construire autres e séries de Puiseux en remplaçant $x^{1/e}$ par $\zeta^k x^{1/e}$.
Elles sont toutes équivalentes et représentées par une seule

Expansion de Puiseux Rationnelle : un couple $(X(t), Y(t)) = (\gamma t^e, \sum_{j=n}^{\infty} \beta_j t^j)$

Expansions de Puiseux Rationnelles de F	$\Longleftrightarrow$	places de la courbe $F(x, y) = 0$

Example (continue): $\rightsquigarrow (X(t), Y(t)) = (t^2, t^3)$

Le diviseur d'adjonction

Le diviseur d'adjonction est

$$\mathcal{A} = \sum_{P \in \mathrm{Sing}(\mathcal{C})} - \sum_{\mathcal{P} | P} \mathrm{val}_{\mathcal{P}} \left(\frac{dx}{F_y} \right) \mathcal{P}$$

$$\xrightarrow[\text{expansions de Puiseux rationnelles}]{\text{En utilisant les}} \mathrm{val}_{\mathcal{P}} \left(\frac{dx}{F_y} \right) = \mathrm{val}_t \left(\frac{et^{e-1}}{F_y(X(t), Y(t), 1)} \right)$$

Le diviseur d'adjonction

Le diviseur d'adjonction est

$$\mathcal{A} = \sum_{P \in \mathrm{Sing}(\mathcal{C})} - \sum_{\mathcal{P}|P} \mathrm{val}_{\mathcal{P}} \left(\frac{dx}{F_y} \right) \mathcal{P}$$

$$\xrightarrow[\text{expansions de Puiseux rationnelles}]{\text{En utilisant les}} \mathrm{val}_{\mathcal{P}} \left(\frac{dx}{F_y} \right) = \mathrm{val}_t \left(\frac{et^{e-1}}{F_y(X(t), Y(t), 1)} \right)$$

Exemple

On considère $\mathcal{C} : y^2 - x^3 = 0$.
L'unique point singulier (non–ordinaire) est $(0, 0)$.

$$(X(t), Y(t)) = (t^2, t^3) \rightsquigarrow \mathrm{val}_{\mathcal{P}} \left(\frac{dx}{F_y} \right) = \mathrm{val}_t \left(\frac{2t}{2t^3} \right) = -2$$

Calcul : algorithme pour les séries de Puiseux de germes de courbes[5]
$\rightsquigarrow \mathcal{A}$ calculé avec $\tilde{O}(\delta^3)$ opérations

[5] A. Poteaux et M. Weimann, Annales Herni Lebesgue, 2021

Sketch de l'algorithme

Input

$\mathcal{C} : F(X, Y, Z) = 0$ *une courbe plane projective,* D *un diviseur lisse.*

Étape 1 : Calcule le diviseur d'adjonction $\mathcal{A}$ ✓ $\leftarrow \tilde{O}(\delta^3)$

Étape 2 : Calcule le dénominateur commun H

Étape 3 : Calcule $(H) - D$ ✓ $\leftarrow \tilde{O}((\delta^2 + \deg D)^2)$

Étape 4 : Calcule les numérateurs G_i (proche de l'étape 2)

Output

Une base de l'espace de Riemann–Roch $L(D)$ *en termes de* H *et des* G_i.

Sketch de l'algorithme

Input

$\mathcal{C} : F(X, Y, Z) = 0$ *une courbe plane projective,* D *un diviseur lisse.*

Étape 1 : Calcule le diviseur d'adjonction $\mathcal{A}$ ✓ $\leftarrow \tilde{O}(\delta^3)$

Étape 2 : Calcule le dénominateur commun H

Étape 3 : Calcule $(H) - D$ ✓ $\leftarrow \tilde{O}((\delta^2 + \deg D)^2)$

Étape 4 : Calcule les numérateurs G_i (proche de l'étape 2)

Output

Une base de l'espace de Riemann–Roch $L(D)$ *en termes de* H *et des* G_i.

Trouver un dénominateur en pratique

Algèbre linéaire classique

Soit $d = \deg H$.

Condition $(H) \geqslant \mathcal{A} + D$

$\rightsquigarrow$ système linéaire avec $\deg \mathcal{A} + \deg D \sim \delta^2 + \deg D$ équations

$\rightsquigarrow$ l'élimination de Gauss coûte

$$\tilde{O}((d\delta + \delta^2 + \deg D)^\omega) \text{ opérations}^6$$

[6] $2 \leqslant \omega \leqslant 3$ est un exposant faisable pour l'algèbre linéaire ($\omega = 2.373$)

Trouver un dénominateur en pratique

Algèbre linéaire classique

Soit $d = \deg H$.

Condition $(H) \geqslant \mathcal{A} + D$

$\rightsquigarrow$ système linéaire avec $\deg \mathcal{A} + \deg D \sim \delta^2 + \deg D$ équations

$\rightsquigarrow$ l'élimination de Gauss coûte

$$\tilde{O}((d\delta + \delta^2 + \deg D)^\omega) \text{ opérations}^6$$

Quelle taille a d?

On montre que $d = \left\lceil \frac{(\delta-1)(\delta-2)+\deg D}{\delta} \right\rceil$ est suffisant

$$\rightsquigarrow \tilde{O}((\delta^2 + \deg D)^\omega) \text{ opérations}$$

[6] $2 \leqslant \omega \leqslant 3$ est un exposant faisable pour l'algèbre linéaire ($\omega = 2.373$)

Deuxième méthode : algèbre linéaire structurée

$$\mathrm{val}_t(H(X(t), Y(t), 1) \geqslant \mathrm{val}_t \left(\frac{et^{e-1}}{F_y(X(t), Y(t), 1)} \right)$$

$\rightsquigarrow$ l'espace de polynômes $H(x, y)$ qui satisfont ces conditions est un $\mathbb{K}[x]$-module

$\rightsquigarrow$ calculer une base[7] coûte $\tilde{O}((\delta^2 + \deg D)^\omega)$ opérations[8]

[7] C.-P. Jeannerod, V. Neiger, É. Schost et G. Villard, Journal of Symbolic Computation, 2017

[8] $2 \leqslant \omega \leqslant 3$ est un exposant faisable pour l'algèbre linéaire ($\omega = 2.373$)

Deuxième méthode : algèbre linéaire structurée

$$\mathrm{val}_t(H(X(t), Y(t), 1) \geqslant \mathrm{val}_t\left(\frac{et^{e-1}}{F_y(X(t), Y(t), 1)}\right)$$

$\rightsquigarrow$ l'espace de polynômes $H(x, y)$ qui satisfont ces conditions est un $\mathbb{K}[x]$-module

$\rightsquigarrow$ calculer une base[7] coûte $\tilde{O}((\delta^2 + \deg D)^\omega)$ opérations[8]

L'exposant de complexité est le même mais...

Avantages :

- en général on obtient une base avec une taille de représentation plus petite
- l'exposant de complexité est meilleur sur des corps algébriquement clos
- on peut s'attendre à des améliorations dans le futur

[7] C.-P. Jeannerod, V. Neiger, É. Schost et G. Villard, Journal of Symbolic Computation, 2017

[8] $2 \leqslant \omega \leqslant 3$ est un exposant faisable pour l'algèbre linéaire ($\omega = 2.373$)

Sketch de l'algorithm

Input

$\mathcal{C} : F(X, Y, Z) = 0$ *une courbe plane projective,* D *un diviseur lisse.*

Étape 1 : Calcule le diviseur d'adjonction $\mathcal{A}$ ✓ $\leftarrow \tilde{O}(\delta^3)$

Étape 2 : Calcule le dénominateur commun H ✓ $\leftarrow \tilde{O}((\delta^2 + \deg D)^\omega)$

Étape 3 : Calcule $(H) - D$ ✓ $\leftarrow \tilde{O}((\delta^2 + \deg D)^2)$

Étape 4 : Calcule les numérateurs G_i (proche de l'étape 2)

Output

Une base de l'espace de Riemann–Roch $L(D)$ *en termes de* H *et des* G_i.

Sketch de l'algorithm

Input

$\mathcal{C} : F(X, Y, Z) = 0$ *une courbe plane projective,* D *un diviseur lisse.*

Étape 1 : Calcule le diviseur d'adjonction $\mathcal{A}$ ✓ $\leftarrow \tilde{O}(\delta^3)$

Étape 2 : Calcule le dénominateur commun H ✓ $\leftarrow \tilde{O}((\delta^2 + \deg D)^\omega)$

Étape 3 : Calcule $(H) - D$ ✓ $\leftarrow \tilde{O}((\delta^2 + \deg D)^2)$

Étape 4 : Calcule les numérateurs G_i ✓ $\leftarrow \tilde{O}((\delta^2 + \deg D)^\omega)$

Output

Une base de l'espace de Riemann–Roch $L(D)$ *en termes de* H *et des* G_i.

Théorème (Abelard, B., Couvreur, Lecerf)

Algorithme de type Las Vegas qui calcule $L(D)$ *en* $\tilde{O}((\delta^2 + \deg D)^\omega)$ *opérations*[9]

[9] avec $2 \leqslant \omega \leqslant 3$ est un exposant faisable pour l'algèbre linéaire ($\omega = 2.373$)

Questions futures

Calcul d'espaces de Riemann–Roch de courbes.

- Implementation de l'algorithme (en cours)
- Calcul d'espaces de Riemann–Roch de courbes non-ordinaires en caractéristique positive "petite" (en cours)
- Améliorer l'exposant de complexité dans le cas non–ordinaire (sous–quadratique ?)

Questions futures

Calcul d'espaces de Riemann–Roch de courbes.

- Implementation de l'algorithme (en cours)
- Calcul d'espaces de Riemann–Roch de courbes non-ordinaires en caractéristique positive "petite" (en cours)
- Améliorer l'exposant de complexité dans le cas non–ordinaire (sous–quadratique ?)

Codes AG en dimension superièure.

- Calcul d'espace de Riemann–Roch de surfaces
  $\rightsquigarrow$ construction de codes AG sur les surfaces

Questions futures

Calcul d'espaces de Riemann–Roch de courbes.

- Implementation de l'algorithme (en cours)
- Calcul d'espaces de Riemann–Roch de courbes non-ordinaires en caractéristique positive "petite" (en cours)
- Améliorer l'exposant de complexité dans le cas non–ordinaire (sous–quadratique ?)

Codes AG en dimension superièure.

- Calcul d'espace de Riemann–Roch de surfaces
  $\rightsquigarrow$ construction de codes AG sur les surfaces

Codes en métrique rang.

- Peut-on utiliser les courbes et leurs espaces de Riemann–Roch pour construire des bons codes en métrique rang ?

Merci de votre attention !

Des questions?

elena.berardini@telecom-paris.fr