Allure of Sage
to Algebraic Coding Theorists and You

Kwankyu Lee

Chosun University, Gwangju, Korea

Universitat Rovira i Virgili

Barcelona, Catalonia

19 Juny, 2017

Abstract

The algebraic function fields part of Sage was greatly enhanced by the recent implementation of global function fields machinery into Sage. The coding theoretic motivation behind this development is presented.

Overview

Decoding Problem of Algebraic Geometry codes

- Reed-Solomon codes are error correcting codes used in Bluray disks.
- Reed-Solomon codes are decoded by the Berlekamp-Massey algorithm.
- AG codes are a vast generalization of Reed-Solomon codes, invented by Goppa in 1980s and extensively studied up to now.

The Berlekamp-Massey algorithm has recently been generalized for all AG codes:

K. Lee, M. Bras-Amorós, and M. E. O'Sullivan, "Unique decoding of general AG codes," *IEEE Trans. Inf. Theory*, vol. 60, no. 4, pp. 2038–2053, 2014.

K. Lee, "Decoding of differential AG codes," *Advances in Mathematics of Communications*, vol. 10, no. 2, pp. 307–319, 2016.

While being proud of the works,

I felt *uneasy* somewhere in my heart...

Algebraic Geometry Codes

- Let X be an algebraic curve over a finite field $\mathbb{F}$ of genus g.
- Let $\mathbb{F}(X)$ denote the function field of X.
- Let Ω_X denote the module of differentials of X.
- Let $P_1, P_2, \ldots, P_n$ be rational places on X, and $D = P_1 + P_2 + \cdots + P_n$.
- For a divisor G on X, let $\mathcal{L}(G) = \{f \in \mathbb{F}(X) \mid (f) + G \geq 0\}$ and $\Omega(G) = \{\omega \in \Omega_X \mid (\omega) \geq G\}$.

Evaluation AG code:

$$C_{\mathcal{L}}(D, G) = \{(f(P_1), f(P_2), \ldots, f(P_n)) \mid f \in \mathcal{L}(G)\}$$

Differential AG code:

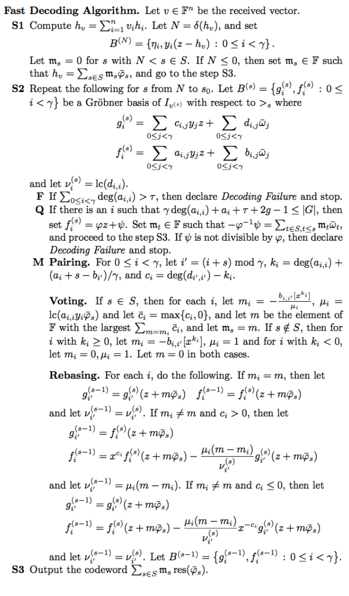
$$C_{\Omega}(D, G) = \{(\mathrm{res}_{P_1}(\omega), \mathrm{res}_{P_2}(\omega), \ldots, \mathrm{res}_{P_n}(\omega)) \mid \omega \in \Omega(-D + G)\}$$

Theorem (Riemann-Roch)

Let A be a divisor on X. Then

$$\dim \mathcal{L}(A) = \deg(A) + 1 - g + \dim \Omega(A).$$

The Decoding Algorithm



Fast Decoding Algorithm. Let $v \in \mathbb{F}^n$ be the received vector.

S1 Compute $h_v = \sum_{i=1}^n v_i h_i$. Let $N = \delta(h_v)$, and set

$$B^{(N)} = \{\eta_i, y_i(z - h_v) : 0 \le i < \gamma\}.$$

Let $\mathfrak{m}_s = 0$ for s with $N < s \in S$. If $N \le 0$, then set $\mathfrak{m}_s \in \mathbb{F}$ such that $h_v = \sum_{s \in S} \mathfrak{m}_s \bar{\varphi}_s$, and go to the step S3.

S2 Repeat the following for s from N to s_0. Let $B^{(s)} = \{g_i^{(s)}, f_i^{(s)} : 0 \le i < \gamma\}$ be a Gröbner basis of $I_{v^{(s)}}$ with respect to $>_s$ where

$$g_i^{(s)} = \sum_{0 \le j < \gamma} c_{i,j} y_j z + \sum_{0 \le j < \gamma} d_{i,j} \bar{\omega}_j$$
$$f_i^{(s)} = \sum_{0 \le j < \gamma} a_{i,j} y_j z + \sum_{0 \le j < \gamma} b_{i,j} \bar{\omega}_j$$

and let $\nu_i^{(s)} = \mathrm{lc}(d_{i,i})$.

F If $\sum_{0 \le i < \gamma} \deg(a_{i,i}) > \tau$, then declare *Decoding Failure* and stop.

Q If there is an i such that $\gamma \deg(a_{i,i}) + a_i + \tau + 2g - 1 \le |G|$, then set $f_i^{(s)} = \varphi z + \psi$. Set $\mathfrak{m}_t \in \mathbb{F}$ such that $-\varphi^{-1}\psi = \sum_{t \in S, t \le s} \mathfrak{m}_t \bar{\omega}_t$, and proceed to the step S3. If ψ is not divisible by φ, then declare *Decoding Failure* and stop.

M **Pairing.** For $0 \le i < \gamma$, let $i' = (i + s) \bmod \gamma$, $k_i = \deg(a_{i,i}) + (a_i + s - b_{i'})/\gamma$, and $c_i = \deg(d_{i',i'}) - k_i$.

Voting. If $s \in S$, then for each i, let $m_i = -\frac{b_{i,i'}[x^{k_i}]}{\mu_i}$, $\mu_i = \mathrm{lc}(a_{i,i} y_i \bar{\varphi}_s)$ and let $\bar{c}_i = \max\{c_i, 0\}$, and let m be the element of $\mathbb{F}$ with the largest $\sum_{m = m_i} \bar{c}_i$, and let $\mathfrak{m}_s = m$. If $s \notin S$, then for i with $k_i \ge 0$, let $m_i = -b_{i,i'}[x^{k_i}]$, $\mu_i = 1$ and for i with $k_i < 0$, let $m_i = 0, \mu_i = 1$. Let $m = 0$ in both cases.

Rebasing. For each i, do the following. If $m_i = m$, then let

$$g_{i'}^{(s-1)} = g_{i'}^{(s)}(z + m\bar{\varphi}_s) \quad f_i^{(s-1)} = f_i^{(s)}(z + m\bar{\varphi}_s)$$

and let $\nu_{i'}^{(s-1)} = \nu_{i'}^{(s)}$. If $m_i \ne m$ and $c_i > 0$, then let

$$g_{i'}^{(s-1)} = f_i^{(s)}(z + m\bar{\varphi}_s)$$
$$f_i^{(s-1)} = x^{c_i} f_i^{(s)}(z + m\bar{\varphi}_s) - \frac{\mu_i(m - m_i)}{\nu_{i'}^{(s)}} g_{i'}^{(s)}(z + m\bar{\varphi}_s)$$

and let $\nu_{i'}^{(s-1)} = \mu_i(m - m_i)$. If $m_i \ne m$ and $c_i \le 0$, then let

$$g_{i'}^{(s-1)} = g_{i'}^{(s)}(z + m\bar{\varphi}_s)$$
$$f_i^{(s-1)} = f_i^{(s)}(z + m\bar{\varphi}_s) - \frac{\mu_i(m - m_i)}{\nu_{i'}^{(s)}} x^{-c_i} g_{i'}^{(s)}(z + m\bar{\varphi}_s)$$

and let $\nu_{i'}^{(s-1)} = \nu_{i'}^{(s)}$. Let $B^{(s-1)} = \{g_i^{(s-1)}, f_i^{(s-1)} : 0 \le i < \gamma\}$.

S3 Output the codeword $\sum_{s \in S} \mathfrak{m}_s \,\mathrm{res}(\bar{\varphi}_s)$.

FIGURE 1. Fast decoding algorithm for differential AG codes

- The algorithm works iteratively with a matrix of univariate polynomials. So it is not at all difficult to implement the algorithm.
- What is difficult is to get the auxiliary data to initialize the algorithm!
- The auxiliary data is obtained by computations in the function field $\mathbb{F}(X)$.

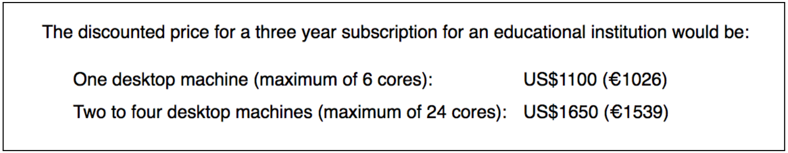
Magma

Few of those who might want to implement the decoding algorithm would have access to computational tools to work with function fields.

Because there is only one computer algebra system in the world that can compute with function fields: Magma.

While writing the papers, the author completely relied on Magma for experiments, which is a *magical black box*.

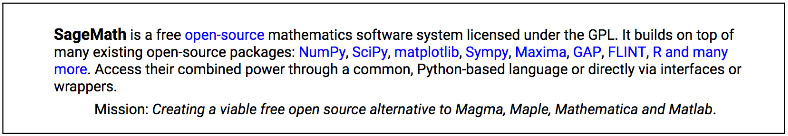
But Magma is expensive.



The discounted price for a three year subscription for an educational institution would be:

One desktop machine (maximum of 6 cores):	US$1100 (€1026)
Two to four desktop machines (maximum of 24 cores):	US$1650 (€1539)

Sage



SageMath is a free open-source mathematics software system licensed under the GPL. It builds on top of many existing open-source packages: NumPy, SciPy, matplotlib, Sympy, Maxima, GAP, FLINT, R and many more. Access their combined power through a common, Python-based language or directly via interfaces or wrappers.

Mission: *Creating a viable free open source alternative to Magma, Maple, Mathematica and Matlab*.

- Sage is like a book written collaboratively.
- If you want, you can examine into the last bit of the internals of Sage.
- If you need, you can fix, enhance, and extend Sage.
- But there is only limited functionality in Sage about function fields.

I had been thinking of implementing global function fields in Sage for many years, but never dared to start.

Finally, I decided to start the task during my sabbatical year,

and completed the task, to my astonishment!

Global Function Fields in Sage

- Function field of the Klein quartic $y^3 + x^3y + x$ over $\mathbb{F}_{16}$

```
sage: K.<x> = FunctionField(GF(16))
sage: R.<Y> = K[]
sage: L.<y> = K.extension(Y^3 + x^3*Y + x)
sage: L
Function field in y defined by y^3 + x^3*y + x
```

- Genus

```
sage: L.genus()
3
```

- Rational places

```
sage: pls = L.places()
sage: pls
[Place (1/x, 1/x^3*y^2 + 1/x),
 Place (1/x, 1/x^3*y^2 + 1/x^2*y + 1),
 Place (x, y),
 ...
 Place (x + z4^2 + z4 + 1, y + z4^3 + z4^2 + 1),
 Place (x + z4^3 + z4^2 + z4 + 1, y + z4^2 + z4)]
sage: len(pls)
17
```

Global Function Fields in Sage

- Divisor A

```
sage: A = 3*pls[0] - 2*pls[2]
sage: A
3*Place (1/x, 1/x^3*y^2 + 1/x) - 2*Place (x, y)
```

- Function space $\mathcal{L}(A)$

```
sage: A.basis_function_space()
[y^2 + x^3]
```

- Differential space $\Omega(A)$

```
sage: A.basis_differential_space()
[(1/x*y^2) d(x), (1/x^2*y^2) d(x)]
```

- Riemann-Roch theorem

```
sage: dimLA = len(A.basis_function_space())
sage: dimOA = len(A.basis_differential_space())
sage: dimLA == A.degree() + 1 - L.genus() + dimOA
True
```

Cartier Codes

Suppose X is a curve over $\mathbb{F}_q$ with $q = p^l$. For a differential $\omega \in \Omega_X$, the Cartier operator is defined by

$$C(\omega) = f_{p-1}\,dx$$

if $\omega = (f_0^p + f_1^p x + \cdots + f_{p-1}^p x^{p-1})dx$ where x is a separating element of the function field.
An important property of the Cartier operator is that

$$\mathrm{res}_P(C(\omega))^p = \mathrm{res}_P(\omega)$$

for any place P and differential ω.
Cartier codes defined by

$$\mathrm{Car}(D,G) = \left\{(\mathrm{res}_{P_1}(\omega), \dots, \mathrm{res}_{P_n}(\omega)) \mid \omega \in \Omega(-D+G)^{C^l}\right\} \subset C_\Omega(D,G)_{|\mathbb{F}_p}$$

generalize the classical Goppa code, which is the workhorse of the McEliece cryptosystem for postquantum cryptography.

A. Couvreur, "Codes and the Cartier operator," *Proceedings of the American Mathematical Society*, vol. 142, no. 6, pp. 1983–1996, 2014.

Cartier Codes

- Differential

```
sage: w = 1/y * y.differential()
sage: w
(x*y + 1/x) d(x)
```

- Divisor of the differential

```
sage: w.divisor().support()
[Place (1/x, 1/x^3*y^2 + 1/x),
 Place (1/x, 1/x^3*y^2 + 1/x^2*y + 1),
 Place (x, y)]
```

- Residues of the differential

```
sage: [w.residue(p) for p in w.divisor().support()]
[0, 1, 1]
```

- Cartier operator

```
sage: w.cartier()
(x*y + 1/x) d(x)
sage: w.cartier() == w
True
```

Cartier Codes

- Klein quartic over $\mathbb{F}_{32}$

```
sage: k.<a> = GF(32)
sage: K.<x> = FunctionField(k); _.<Y> = K[]
sage: L.<y> = K.extension(Y^3 + x^3*Y + x)
```

- Divisors D, G

```
sage: pls1 = L.places(1)
sage: D = sum(pls1)
sage: Z = L.places_finite(2)[-1]
sage: G = 2*Z
```

- Load the code for Cartier codes

```
sage: load('cartier_code.py')
```

- Fire the code!

```
sage: code = cartier_code(D,G)
sage: code
[33, 21] linear code over GF(2)
sage: code.minimum_distance()
6
```

This code has the best minimum distance for the length and dimension!

Weierstrass Numerical Semigroup

- Gaps for ordinary places of the Klein quartic over $\mathbb{F}_2$

  ```
  sage: L.gaps()
  [1, 2, 3]
  ```

  Thus the numerical semigroup for an ordinary place is $0, 4, 5, 6, 7, 8, 9, \dots$.

- Gaps for Weierstrass places of the Klein quartic over $\mathbb{F}_2$

  ```
  sage: wpls = L.weierstrass_places()
  sage: [(p.gaps(),p) for p in wpls]
  [([1, 2, 4], Place (1/x, 1/x^3*y^2 + 1/x)),
   ([1, 2, 4], Place (1/x, 1/x^3*y^2 + 1/x^2*y + 1)),
   ([1, 2, 4], Place (x, y)),
   ([1, 2, 4], Place (x + 1, (x^3 + 1)*y + x + 1)),
   ([1, 2, 4], Place (x^3 + x + 1, y + 1)),
   ([1, 2, 4], Place (x^3 + x + 1, y + x^2)),
   ([1, 2, 4], Place (x^3 + x + 1, y + x^2 + 1)),
   ([1, 2, 4], Place (x^3 + x^2 + 1, y + x)),
   ([1, 2, 4], Place (x^3 + x^2 + 1, y + x^2 + 1)),
   ([1, 2, 4], Place (x^3 + x^2 + 1, y + x^2 + x + 1))]
  ```

  The numerical semigroup for a Weierstrass place is $0, 3, 5, 6, 7, 8, 9, \dots$.

Feng-Rao Numbers of a Numerical Semigroup

Let Γ be a numerical semigroup for place Q. For $r \geq 1$, the rth Feng-Rao distance is defined to be

$$\delta_r(m) = \min\{|d(a_1, a_2, \dots, a_r)| \mid m < a_1 < a_2 < \cdots < a_r\}$$

where $d(a_1, a_2, \dots, a_r) = \bigcup_{i=1}^{r}\{a \in \Gamma \mid a_i - a \in \Gamma\}$. It is known that for $r \geq 1$, $d_r(C_\Omega(D, mQ)) \geq \delta_r(m)$ and there exists a nonnegative integer $E(\Gamma, r)$ such that

$$\delta_r(m) = m + 2 - 2g + E(\Gamma, r)$$

for all $m \geq 2c - 2$ with conductor c of Γ. The number $E(\Gamma, r)$ is called the rth Feng-Rao number of the numerical semigroup Γ.

Example (Klein quartic over $\mathbb{F}_2$)

Feng-Rao numbers for the numerical semigroup at the ordinary places

$$0, 2, 4, 6, 7, 8, 9, 10, 11, \dots$$

Feng-Rao numbers for the numerical semigroup at the Weierstrass places

$$0, 3, 4, 6, 7, 8, 9, 10, 11, \dots$$

How to Get the Code

- Get Sage

  `http://www.sagemath.org`
- Checkout ticket 22982

  `https://trac.sagemath.org/ticket/22982`
- Build Sage

  `http://doc.sagemath.org/html/en/developer`
- Read the documentation

Enjoy! I welcome your feedback.

Moltes gràcies