Proceedings of the Jangjeon Mathematical Society
20 (2017), No. 2. pp. 203 - 211

www.jangjeon.or.kr
http://dx.doi.org/10.23001/pjms2017.20.2.203

COMPUTING INITIAL DATA FOR DECODING ALGORITHMS FOR GENERAL AG CODES

KWANKYU LEE

Abstract. An efficient algorithm decoding Goppa's codes on algebraic curves over finite fields, called AG codes, appeared only recently. Each instance of the decoding algorithm for a specific AG code requires some precomputed initial data about the Riemann-Roch spaces of either functions or differentials of the given curve. As Magma is particularly good at computing with these spaces, we present the details of the Magma implementation of the procedure computing the initial data for the decoding algorithm.

1. Introduction

Let X be a smooth geometrically irreducible projective curve defined over a finite field $\mathbb{F}$ of genus g. Let $\mathbb{F}(X)$ and Ω_X denote the function field and the module of differentials of X respectively. Let $P_1, P_2, \dots, P_n$ be distinct rational points on X, and $D = P_1 + P_2 + \cdots + P_n$. Let G be an arbitrary divisor on X, whose support does not contain the rational points. Recall that $\mathcal{L}(G) = \{f \in \mathbb{F}(X) \mid (f) + G \geq 0\}$ and $\Omega(G) = \{\omega \in \Omega_X \mid (\omega) \geq G\}$. Goppa [7] defined two kinds of error correcting codes

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

and

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

which are respectively called an evaluation AG code and a differential AG code. As well known, they are dual to each other.

AG codes provide series of codes with large minimum distances surpassing the Gilbert-Varshamov bound [14]. Research on the dimension and the minimum distance of AG codes has deep connections with important problems of discrete mathematics, number theory and algebraic geometry [13]. Moreover, with the projective line taken for X, the class of AG codes and their subfield subcodes includes Reed-Solomon codes and BCH codes that have been used in various communication and storage devices. Goppa codes used in the cryptosystem proposed by McEliece are subfield subcodes of differential AG codes on the projective line. Generalizing decoding algorithms for Reed-Solomon and BCH codes, efficient decoding algorithms

2010 *Mathematics Subject Classification.* Primary 94B35, 94B27, 68W30.

Key words and phrases. Decoding algorithm, Algebraic Geometry codes, Algebraic curves.

This is an expanded version of the paper presented in the International Congress on Mathematical Software 2014, whose extended abstract was published in the conference proceeding [8]. The main result only appears in the present paper, in Section 3.

K. Lee is with the Department of Mathematics Education, Chosun University, Gwangju 61452, Korea (e-mail: kwankyu@chosun.ac.kr). This study was supported by research fund from Chosun University, 2016.

for a subclass of AG codes have been devised. Some AG codes are strong candidates to replace Reed-Solomon codes. AG codes and their subfield subcodes on curves of positive genus are considered for the McEliece cryptosystem and for secret sharing schemes. For research on and practice of these applications, availability of efficient decoding algorithms for AG codes is essential.

Then it is unfortunate to find that decoding algorithms for AG codes are currently very poorly available in computer algebra systems. In Singular and Magma, Skorobogatov and Vladut's basic algorithm for general differential AG codes [14] is implemented, but this is almost all one can find in public. The basic algorithm requires as input another divisor G_1 such that the supports of G_1 and D are disjoint and

$$\deg(G_1) < \deg(G) - 2g + 2 - t \quad \text{and} \quad \dim_{\mathbb{F}}(\mathcal{L}(G_1)) > t \tag{1}$$

to correct all errors of weight $\leq t$. Though it is known that G_1 exists for all $t \leq (d^* - 1 - g)/2$, where $d^* = \deg G - 2g + 2$ is the Goppa bound on the minimum distance or the designed distance of $C_\Omega(D, G)$, users of Singular and Magma have to provide G_1 to run the basic algorithm [3, 2].

However, in the research literature, actually there have been much more advances on the problem of decoding AG codes. By extensive works of Feng, Rao, Sakata and many others, a fast decoding algorithm that corrects errors of weight less than half of d^* for one-point differential AG codes, in which $G = mQ$, was already established in 1990's [6, 11]. Moreover, decoding algorithms for multi-point differential and evaluation AG codes have been proposed by Duursma [4], Beelen and Høholdt [1], and Sakata and Fujisawa [12]. Unfortunately these algorithms were never available in public on computer algebra systems. This situation may be due to that none of these algorithms are fast, simple to implement, and easy to apply for general AG codes.

Recently, improving the current situation, there appeared a fast decoding algorithm for general evaluation and differential AG codes [10, 9] that can correct up to half of their designed distances.[1] This algorithm is very simple and easy to implement. Indeed all heavy computations are done just to provide initial data to the algorithm, and for each received vector, the algorithm performs simple iterative procedure to recover message symbols. Computations of the initial data are all ultimately based on computations of a basis of the Riemann-Roch space $\mathcal{L}(G)$ or $\Omega(G)$ for any divisor G. As Magma has very nice facilities for computing with these spaces, the decoding algorithm can be most easily implemented in Magma. In the next section, we show a Magma session with a decoding example from the Magma documentation.

2. Example Session

The following Magma scripts construct $[23, 14, 7]$ differential AG code C on the Klein quartic, of genus $g = 3$, defined over $\mathbb{F}_8$ on the projective plane.

```
> F<a> := GF(8);
> PS<x,y,z> := ProjectiveSpace(F, 2);
> Cv := Curve(PS, x^3*y + y^3*z + x*z^3);
> Pl := Places(Cv, 1); // rational points
```

[1]Precisely speaking, general AG codes mean multi-point AG codes, which allows arbitrary G but requires a rational point Q not in the support of D.

```
> Q := Place(Cv![0,1,0]);
> P := [Pl[i]: i in [1..#Pl] | Pl[i] ne Q];
> G := 11*Q;
> C := AGDualCode(P, G);
```

With the rational point $Q = [0 : 1 : 0]$ on the curve, the divisor $G_1 = 4Q$ satisfies the condition (1) for $t = 1$. Thus the Magma intrinsic AGDecode, which implements Skorobogatov and Vladut's basic algorithm, can correct arbitrary errors of weight 1.

```
> v := Random(C);
> rec_vec := v;
> rec_vec[Random(1,Length(C))] +:= Random(F);
> res := AGDecode(C, rec_vec, 4*Q);
> res eq v;
true
```

However, observe that as the minimum distance of C is 7, the code has capability of correcting unambiguously errors of weight at most 3. So the decoding algorithm in Magma fails to exert the full potential of the code.

Now we turn to the new Magma intrinsic DifferentialAGCode, which implements the decoding algorithm for differential AG codes in [9].

```
> D:=&+P;
> code:=DifferentialAGCode(D,G,Q);
> code eq C;
true
> code`DecodingRadius;
3
```

The last output indicates that the code is capable of correcting errors of weight up to half of the actual minimum distance of C.

```
> res := DecodeAGCode(code,rec_vec);
> res eq v;
true
> rec_vec[Random(1,Length(C))] +:= Random(F);
> rec_vec[Random(1,Length(C))] +:= Random(F);
> Distance(rec_vec, v);
3
> res := DecodeAGCode(code,rec_vec);
> res eq v;
true
```

There is also new Magama intrinsic EvaluationAGCode, which implements the new decoding algorithm for evaluation AG code in [10].

```
> ecode := EvaluationAGCode(D,G,Q);
> Dual(ecode) eq code;
true
> ecode`DecodingRadius;
5
> MinimumDistance(ecode);
12
```

Note that `ecode` is a $[23, 9, 12]$ evaluation AG code, dual of `code`, and can correct errors of weight up to 5.

3. Implementation Details

The details of the decoding algorithm implemented in the Magma intrinsic `EvaluationAGCode` is fully described in [10]. The efficiency and simplicity of the decoding algorithm owes largely to the initial data precomputed before actual decoding procedure. The data are just polynomials over the finite field $\mathbb{F}$ representing elements of Riemann-Roch spaces. This polynomial representation is made possible by so-called Apéry systems on the function field. In this section, we explain how to compute the Apéry systems and the initial data for the decoding algorithm. We use the $[23, 9, 12]$ evaluation AG code in the previous section as an example throughout this section.

First we recall some definitions from [10]. Let

$$R = \bigcup_{s=0}^{\infty} \mathcal{L}(sQ) \subset \mathbb{F}(X).$$

For $f \in R$, let $\rho(f) = -v_Q(f)$, that is the smallest s such that $f \in \mathcal{L}(sQ)$. The Weierstrass semigroup at Q is then

$$\Lambda = \{\rho(f) \mid f \in R\} = \{\lambda_0, \lambda_1, \lambda_2, \dots\} \subset \mathbb{Z}_{\geq 0}.$$

It is well known that Λ is a numerical semigroup whose number of gaps is the genus g of X. The nonnegative integers in Λ are called nongaps. Let γ be the smallest positive nongap, and let $\rho(x) = \gamma$ for some $x \in R$.

Recall that for divisor G, the Magma command `Basis(G)` computes a basis of $\mathcal{L}(G)$ and `Dimension(G)` computes the dimension of $\mathcal{L}(G)$. These commands are used to compute γ and x with $\rho(x) = \gamma$ in the following scripts.

```
function get_gamma()
    s:=1;
    while true do
        if Dimension(s*Q) gt 1 then
            return s;
        end if;
        s:=s+1;
    end while;
end function;
gamma:=get_gamma();

function get_xR()
    for b in Basis(gamma*Q) do
        if Valuation(b,Q) eq -gamma then
            return b;
        end if;
    end for;
end function;
xR:=get_xR();
```

For our example, we get

```
> gamma;
3
> xR;
X
```

where `X` and `Y` denote generating elements of the function field of X, which is denoted by `FF`.

For each $0 \leq i < \gamma$, let a_i be the smallest nongap such that $a_i \equiv i \pmod{\gamma}$ and $\rho(y_i) = a_i$ for some $y_i \in R$. Then we can show that every element of R can be written as a unique $\mathbb{F}$-linear combination of the monomials $\{x^k y_i \mid k \geq 0, 0 \leq i < \gamma\}$. The set $\{y_i \mid 0 \leq i < \gamma\}$ is called an Apéry system of R. The following scripts compute the Apéry system of R.

```
function apery_R()
    s:=0;
    dR:=[0:i in [1..gamma]];
    yR:=[FF|0:i in [1..gamma]];
    found:=[false:i in [1..gamma]];
    num:=0;
    while num lt gamma do
        B:=Basis(s*Q);
        g:=0;
        for b in B do
            if Valuation(b,Q) eq -s then
                g:=b;
                break;
            end if;
        end for;
        r:=(s mod gamma)+1;
        if g ne 0 and not found[r] then
            dR[r]:=s;
            yR[r]:=g;
            found[r]:=true;
            num+:=1;
        end if;
        s:=s+1;
    end while;
    return dR,yR;
end function;
dR,yR:=apery_R();
```

For our example, we get

```
> dR;
[ 0, 7, 5 ]
> yR;
[
    1,
    X*Y^2,
    X*Y
]
```

Let

$$\bar{R} = \bigcup_{s=-\infty}^{\infty} \mathcal{L}(sQ+G) \subset \mathbb{F}(X),$$

which is clearly a module over R. For $f \in \bar{R}$, let $\delta(f) = -v_Q(f) - v_Q(G)$, that is also the smallest integer s such that $f \in \mathcal{L}(sQ+G)$. Let

$$\bar{\Lambda} = \{\delta(f) \mid f \in \bar{R}\} = \{s_0, s_1, s_2, \dots\} \subset \mathbb{Z}_{\geq -\deg(G)}.$$

Note that $\Lambda + \bar{\Lambda} = \bar{\Lambda}$, and in this sense $\bar{\Lambda}$ is a numerical Λ-module. The integers in $\bar{\Lambda}$ are called nongaps. For each $0 \leq i < \gamma$, there exists the smallest nongap b_i of $\bar{\Lambda}$ such that $b_i \equiv i \pmod{\gamma}$ and $\delta(\bar{y}_i) = b_i$ for some $\bar{y}_i \in \bar{R}$. We can still show that $\{\bar{y}_i \mid 0 \leq i < \gamma\}$ forms a basis of $\bar{R}$ as a free module of rank γ over $\mathbb{F}[x]$. For $s \in \bar{\Lambda}$, define $\bar{\varphi}_s = x^k \bar{y}_i$ for $i = s \mod \gamma$ and $k = (s - b_i)/\gamma \geq 0$. Then $\delta(\bar{\varphi}_s) = s$. Thus $\{\bar{\varphi}_s \mid s \in \bar{\Lambda}\}$ is a basis of $\bar{R}$ over $\mathbb{F}$, members of which are called the monomials of $\bar{R}$. The set $\{\bar{y}_i \mid 0 \leq i < \gamma\}$ is called the Apéry system of $\bar{R}$.

As the definitions are similar, the computation of the Apéry system of $\bar{R}$ is almost identical with that of R. So we omit the scripts for the computation. For our example, we get the following output from the scripts.

```
> dRbar;
[ -6, -11, -4 ]
> yRbar;
[
    X*Y,
    1,
    X*Y^2
]
>
```

Now that the Apéry system of $\bar{R}$ is available, an element f of $\bar{R}$ can be represented as a vector $(c_0, c_1, \dots, c_{\gamma-1}) \in \mathbb{F}[x]^\gamma$ if $f = \sum_{i=0}^{\gamma-1} c_i \bar{y}_i$. The following script computes the vector form of the given f.

```
function vec_form(f)
    r:=f;
    l:=[W|0:i in [1..gamma]];
    while r ne 0 do
        s:=-Valuation(r,Q)-Valuation(G,Q);
        e:=exponents(s);
        mon:=xR^e[1]*yRbar[e[2]+1];
        c:=Evaluate(r/mon,Q);
        l[e[2]+1]+:=c*x^e[1];
        r-:=c*mon;
    end while;
    return Vector(l);
end function;
```

where `W` denotes the polynomial ring $\mathbb{F}[x]$ of Magma with variable `x`, with which the decoding algorithm *w*orks.

The evaluation map

$$\text{ev} : \bar{R} \to \mathbb{F}^n, \quad \varphi \mapsto (\varphi(P_1), \varphi(P_2), \dots, \varphi(P_n))$$

is linear over $\mathbb{F}$. The evaluation AG code is now simply $C_{\mathcal{L}}(D,G) = \mathrm{ev}(\mathcal{L}(G))$, that is the image of $\mathcal{L}(G)$ under the evaluation map. The map ev is surjective onto $\mathbb{F}^n$. Let $h_i \in \bar{R}$ be such that $\mathrm{ev}(h_i)$ is the ith element of the standard basis of $\mathbb{F}^n$. Let J be the kernel of ev. Note that J is a submodule of $\bar{R}$ over R, and also over $\mathbb{F}[x]$. Let $\{\eta_i \mid 0 \le i < \gamma\}$ be a Gröbner basis of J over $\mathbb{F}[x]$ such that $\deg_{\bar{y}}(\mathrm{lt}(\eta_i)) = i$. We may call h_i Lagrange basis polynomials as their $\mathbb{F}$-linear combinations yield functions in $\bar{R}$ that correspond to vectors of $\mathbb{F}^n$ under ev. The Lagrange basis polynomials and the Gröbner basis of J are essential initial data supplied to the decoding algorithm before actual decoding processing. To compute them, we deploy an FGLM-like algorithm [5].

```
function get_eta_basis()
    basis:=[];
    found:=[false:i in [1..gamma]];
    s:=s0;
    mat:=Matrix(ev_mon(s));
    delta:=[exponents(s)];
    num:=0;
    while num lt gamma do
        s:=next(s);
        e:=exponents(s);
        if not found[e[2]+1] then
            v:=ev_mon(s);
            ans,sol:=IsConsistent(mat,v);
            if ans then
                gen:=[W!0 : i in [1..gamma]];
                for i in [1..#delta] do
                    gen[delta[i][2]+1]+:=-sol[i]*x^delta[i][1];
                end for;
                gen[e[2]+1]+:=x^e[1];
                basis[e[2]+1]:=Vector(gen);
                found[e[2]+1]:=true;
                num+:=1;
            else
                mat:=VerticalJoin(mat,Matrix(v));
                Append(~delta,e);
            end if;
        end if;
    end while;

    vecs:=[];
    matinv:=mat^-1;
    for i in [1..n] do
        h:=[W!0 : i in [1..gamma]];
        for j in [1..n] do
            h[delta[j,2]+1]+:=matinv[i,j]*x^delta[j,1];
        end for;
        vecs[i]:=Vector(h);
    end for;
    return basis,vecs;
end function;
eta_vecs,hvecs:=get_eta_basis();
```

where `s0` denotes the smallest nongap s_0 in $\bar{\Lambda}$, the command `ev_mon(s)` gives the vector $\mathrm{ev}(\bar{\varphi}_s)$, the command `next(s)` gives the nongap subsequent to s in $\bar{\Lambda}$, the command `exponents(s)` gives the pair (k, i) satisfying $\gamma k + b_i = s$, that is $x^k \bar{y}_i = \bar{\varphi}_s$. For our example, we get

```
> eta_vecs;
[
    (  x^7 + 1         0         0),
    (       0 x^8 + x         0),
    (       0         0 x^8 + x)
]
> hvecs;
[
    (        0   x^7 + 1   x^7 + 1),
    (        0         0   x^7 + 1),
    ...
]
```

Thus $\eta_0 = (x^7{+}1)\bar{y}_0, \eta_1 = (x^8{+}x)\bar{y}_1, \eta_2 = (x^8{+}x)\bar{y}_2$ and $h_1 = (x^7{+}1)\bar{y}_1{+}(x^7{+}1)\bar{y}_2$, and so forth.

The decoding algorithm for differential AG codes, implemented in the Magma intrinsic `DifferentialAGCode`, is a mirror image of its companion algorithm in the sense that the algorithmic structures are the same but the underlying mathematical objects are different. This simply reflects the fact that the space $\Omega(-D+G)$ of differentials and the residue map res defines the differential AG code while the space $\mathcal{L}(G)$ and the evaluation map ev defines the evaluation AG code. Thus instead of $\bar{R}$, we define

$$\overline{W} = \bigcup_{s=-\infty}^{\infty} \Omega(-D+G-sQ) \subset \Omega_X,$$

which is again a module over R, and define the residue map

$$\mathrm{res} : \overline{W} \to \mathbb{F}^n, \quad \omega \mapsto (\mathrm{res}_{P_1}(\omega), \mathrm{res}_{P_2}(\omega), \dots, \mathrm{res}_{P_n}(\omega))$$

which is linear over $\mathbb{F}$. The rest of definitions and computations regarding $\overline{W}$ and res go in parallel with that of $\bar{R}$ and ev. So we leave the reader with the task of writing Magma scripts corresponding to those presented previously. For an exposition and analysis about the decoding algorithm itself, see [9].

Acknowledgement

The author would like to express his gratitude towards Professor Dae San Kim, who as a teacher and advisor shaped the author's career in mathematics. This paper is dedicated to him for his honorable retirement from Sogang University in Seoul, August of 2016.

References

[1] Peter Beelen and Tom Høholdt. The decoding of algebraic geometry codes. In *Advances in algebraic geometry codes*, volume 5 of *Ser. Coding Theory Cryptol.*, pages 49–98. World Sci. Publ., Hackensack, NJ, 2008.

[2] Wieb Bosma, John Cannon, and Catherine Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997.

[3] Wolfram Decker, Gert-Martin Greuel, Gerhard Pfister, and Hans Schönemann. Singular 3-1-6 — A computer algebra system for polynomial computations. `http://www.singular.uni-kl.de`, 2012.

[4] Iwan M. Duursma. Majority coset decoding. *IEEE Trans. Inf. Theory*, 39(3):1067–1070, 1993.

[5] J. C. Faugère, P. Gianni, D. Lazard, and T. Mora. Efficient computation of zero-dimensional Gröbner bases by change of ordering. *J. Symbolic Comput.*, 16(4):329–344, 1993.

[6] Gui Liang Feng and T. T. N. Rao. Decoding algebraic-geometric codes up to the designed minimum distance. *IEEE Trans. Inf. Theory*, 39(1):37–45, 1993.

[7] V. D. Goppa. Codes on algebraic curves. *Sov. Math. Dokl.*, 24(1):170–172, 1981.

[8] Kwankyu Lee. Magma implementation of decoding algorithms for general algebraic geometry codes. In *International Congress on Mathematical Software*, pages 119–123. Springer, 2014.

[9] Kwankyu Lee. Decoding of differential AG codes. *Advances in Mathematics of Communications*, 10(2):307–319, 2016.

[10] Kwankyu Lee, Maria Bras-Amorós, and Michael E. O'Sullivan. Unique decoding of general AG codes. *IEEE Trans. Inf. Theory*, 60(4):2038–2053, 2014.

[11] S. Sakata, H. E. Jensen, and T. Høholdt. Generalized Berlekamp-Massey decoding of algebraic-geometric codes up to half the Feng-Rao bound. *IEEE Trans. Inf. Theory*, 41(6):1762–1768, 1995.

[12] Shojiro Sakata and Masaya Fujisawa. Fast decoding of multipoint codes from algebraic curves. *IEEE Trans. Inf. Theory*, 60(4):2054–2063, 2014.

[13] Serguei A. Stepanov. *Codes on Algebraic Curves*. Springer, 1999.

[14] Henning Stichtenoth. *Algebraic Function Fields and Codes*. Springer-Verlag, second edition, 2009.

Department of Mathematics Education, Chosun University, Gwangju 61452, Korea
E-mail address: `kwankyu@chosun.ac.kr`