A tutorial on AG code construction from a Gröbner basis perspective

Douglas A. Leonard
Department of Mathematics and Statistics
Auburn University
Auburn, AL 36849

1 Introduction

This tutorial is meant to stimulate interplay between those working with AG codes and those working with Gröbner bases, as well as making this material more accessible to those not trained in algebraic geometry. (It will be presented in two separate chapters. This first chapter is meant primarily as an introduction to AG codes, but includes material on producing proper *one-point* descriptions of these codes as well. The second chapter will contain material about *syndrome decoding* and *list decoding*.) The terminology chosen is therefore that of the easily understood concepts of *multivariate polynomial rings* and *ideals* of relations among the variables, which is closer to the notation of *function fields*, and much more useful computationally than the more standard algebraic geometry terminology. It is also an approach that generalizes to other projective varieties. Gröbner basics are covered elsewhere in this volume, but there are sections introducing the needed concepts about weighted orderings and the needed concepts for describing RS and AG codes; given that the readers of this paper may know one topic but not the other, or may know both but have not seen this material expressed in a language that can be used by both. (Those familiar only with RS codes and not AG codes, should be able to catch on by paying attention to the parity-check or generator functions used in the examples, since all the algorithms are generalizations of algorithms known for RS codes.)

History and bibliography, at least through 1998, is sufficiently covered by Høholdt, Pellikaan, and van Lint in chapter 10 of the Handbook of Coding Theory [9], but some interesting papers in the literature about producing one-point descriptions relative to this viewpoint are [5], [6], [7]. There are also some recent papers related to finding good AG codes [1],[4], which in turn cite the literature of that topic. Examples are freely "borrowed" from existing literature because they are described and/or worked differently here, reflecting more or less a decade of extra insight. (Though there is some background material throughout, this is not meant as a "survey" paper, but strictly as a "tutorial" for Gröbner basis methods that can be applied to AG codes to properly define them (this chapter) and decode them (subsequent chapter).)

Linear codes in general are subspaces of *dimension* k of $\mathbf{F}_q^n$, n being the *wordlength*. Such codes are the row-spaces of $k \times n$ *generator matrices* and have $(n-k) \times n$ *parity-check matrices* (generators for the *orthogonal complements* or *dual codes*), given either explicitly or implicitly.

Consider an *evaluation matrix* $Eval$ with entries $Eval_{i,j} := f_i(P_j)$ for some *points* P_j and *functions* f_i. The classic *RS codes* (short for *Reed-Solomon*) have both generator and parity-check matrices of this form, with points P_j the non-zero elements of $\mathbf{F}_q$ (with extended RS codes using 0 as well) and functions $f_i := x^i$ for some appropriate consecutive sequence of indices i.

Functionally decoded RS codes (those with $Eval$ as parity-check matrix) generally use *syndromes* s_i, the entries of $\underline{s} := \underline{r} Eval^T = \underline{e} Eval^T$ for decoding by determining the error $\underline{e}$ with weight less than $d/2$, d the *minimum distance* of the code. Algorithms such as *Berlekamp-Massey* or the *extended Euclidean algorithm* are used to reduce a syndrome matrix efficiently, to produce an *error-locator polynomial*, with roots the *error positions*. There are various other algorithms that can then be invoked to calculate the *error magnitudes*. *Functionally encoded* RS codes (those with $Eval$ as generator matrix) on the other hand, lend themselves to *list-decoding* techniques for recovering a list of messages with some corresponding to codewords $\underline{c}$ close to the received word $\underline{r}$.

RS codes are *maximum-distance-separable* codes, satisfying the *Singleton bound*, $k + d = n + 1$. But the wordlength is bounded by the field size q, so that creating longer length codes requires increasing the field size, or paying a severe penalty to use *subfield subcodes*, as with *BCH* codes.

To generalize these codes to *AG codes* (short for *algebraic geometry codes* in recognition of their origin), first start by thinking of the elements $\alpha \in \mathbf{F}_q$ as *affine points*, then change them to *projective points* $(\alpha : 1)$ on the *projective line* over $\overline{\mathbf{F}}_q$, the *algebraic closure* of $\mathbf{F}_q$; having one extra point $P_\infty := (1 : 0)$, at which the *rational homogeneous functions* $(x_1/x_0)^i$ have all i of their *poles*. Then $Eval_{i,j} = (x_1/x_0)^i((\alpha_j : 1)) = (\alpha_j/1)^i$. (So far this is merely a notational change to motivate the generalization.)

Then consider replacing the projective line by a *projective variety*,

$$\mathcal{X} := \{(x_N : \cdots : x_1 : x_0) \in \mathbf{P}^N(\overline{\mathbf{F}}_q) : f(x_N, \ldots, x_1, x_0) = 0 \text{ for all } f \in \mathcal{I}\}$$

for some *ideal* $\mathcal{I}$ of defining relations (polynomials that should be zero at points of $\mathcal{X}$). $\mathcal{X}$ is a *projective curve*, roughly speaking, when there is only one *independent* variable relative to these relations. Some such curves have many points and easily described parameters.

Simple examples of this are *Hermitian curves* defined projectively by

$$\left(\frac{x_2}{x_0}\right)^q + \left(\frac{x_2}{x_0}\right) - \left(\frac{x_1}{x_0}\right)^{q+1} = 0$$

or in easier affine terms (without the *homogenizing variable* x_0) by $x_2^q + x_2 - x_1^{q+1} = 0$ having $1 + q^3$ points rational over $\mathbf{F}_{q^2}$ There are more complicated examples, such as those from the *towers* defined by Garcia and Stichtenoth [3], described in affine terms by

$$x_{i+1}^q + x_{i+1} = \frac{x_i^q}{x_i^{q-1} + 1},\ 1 \leq i \leq N.$$

(Warning such descriptions are *not* unique, as shall be mentioned in the section on curve definition below.)

These AG codes pay a *penalty*, g, relative to the Singleton bound, namely that $k+d \geq n+1-g$. They were initially of interest as a source of codes with $k/n+d/n \geq 1-(g-1)/n \geq 1-1/(\sqrt{q}-1)$, the *Tsfasman-Vlăduţ-Zink bound*, which can be better than the more traditional *Gilbert-Varshamov bound*, meaning that they had a reasonably good tradeoff between *information rate* and *relative distance*. Moreover, they were codes with structure to them as well as having good parameters, meaning that they probably could be efficiently decoded. While these codes can have length much larger than q, it is at most the *Hasse-Weil bound* $n \leq q + 1 + 2g\sqrt{q}$.

This chapter of the tutorial will discuss producing a reasonable description (in terms of the function space of an evaluation matrix) of AG codes from less useful definitions (as above), will the subsequent chapter will be devoted to syndrome decoding algorithms for functionally decoded AG codes (called *geometric Goppa codes* $\mathcal{C}^*(\mathcal{D}, \mathcal{G})$ in the Handbook of Coding Theory [9]) ; and list-decoding algorithms for functionally encoded AG codes (called *geometric Reed-Solomon codes* $\mathcal{C}(\mathcal{D}, \mathcal{G})$ in the same handbook). All are topics intimately related to *ideals* and their *Gröbner bases* and *Δ-sets* (or *footprints* or *standard monomial bases*). However, in general, in the literature, for various reasons, this purely ideal-theoretic approach is often slighted.

There are example programs for some of this material, written in MAGMA, as opposed to some generic pseudo-code. Those with access to MAGMA can run these easily enough, while those without such access will have to treat it as fairly readable pseudo-code. More example calculations and programs should be available on the author's Auburn University website:

```
www.dms.auburn.edu/~leonada
```

2 Traditional AG approach

A more traditional algebraic geometry approach to curves might start with *divisors*, $\mathcal{D} := \sum_P n_P \cdot P$, with (finite) *degree* $deg(\mathcal{D}) := \sum_P n_P$, which are usefule additive bookkeeping devices for keeping track of zeros and poles of functions. As an example, relative to the Klein quartic, the divisors

$$\left(\frac{x_1}{x_0}\right) = -2\cdot P - 1\cdot Q + 3\cdot R, \quad \left(\frac{x_2}{x_0}\right) = -3\cdot P + 2\cdot Q + 1\cdot R$$

denote that both functions have 3 poles and 3 zeros restricted to the support $P := (1:0:0)$, $Q := (0:1:0)$, and $R := (0:0:1)$. In general, *(equivalence classes of rational homogeneous) functions (modulo the curve)* necessarily have the same (finite) number of poles as zeros, except for the identically zero function itself. More significantly, these functions have *Laurent series expansions* at each point P on the curve, written in terms of some *local parameter*, t_P, a function having a simple zero at P. So, for instance,

$$\frac{x_1}{x_0} = t_P^{-2} + \cdots = t_Q^{-1} = t_R^{3} + \cdots, \quad \frac{x_2}{x_0} = t_P^{-3} + \cdots = t_Q^{2} + \cdots = t_R^{1}$$

if $t_P := x_1/x_2$, $t_Q := x_0/x_1$, and $t_R := x_2/x_0$.

Divisors are also useful in defining vector spaces modulo the curve; namely

$$\mathcal{L}(\mathcal{D}) := \{0\} \cup \{f : (f) + \mathcal{D} \succeq 0\},$$

meaning that the *valuation* $\nu_P(f)$, the *trailing exponent* in the Laurent series expansion of f at P satisfies $\nu_P(f) + n_P(\mathcal{D}) \geq 0$

Differentials, df, can also be defined modulo the curve; and by rewriting $df = f_P dt(P)$, the divisor $(df) := \sum_P \nu_P(f_P)\cdot P$ can be defined as well. If ω is a fixed differential, then the *Riemann-Roch theorem* is:

$$dim(\mathcal{L}(\mathcal{D})) - dim(\mathcal{L}((\omega) - \mathcal{D})) = deg(\mathcal{D}) - g + 1$$

for g the *genus* of the underlying (smooth projective) curve. And the corollary that $deg((\omega)) = 2g-2$ suggests a method of computing said genus.

Here the divisor $\mathcal{D}$ in the definitions of AG codes above will always be of the form $\sum_{j=1}^{n} 1\cdot P_j$, denoting the points used for evaluation. The divisor $\mathcal{G}$, with support disjoint from that of $\mathcal{D}$, will define the vector space $\mathcal{L}(\mathcal{G})$ of functions used for evaluation. And for *one-point AG codes* this means $\mathcal{G} = m\cdot P_\infty$, so that these functions will have at most a pole of order m at P_∞ and no other poles. The code $\mathcal{C}^*(\mathcal{D},\mathcal{G}) = \mathcal{C}(\mathcal{D},(\omega)+\mathcal{D}-\mathcal{G})$, but the encoding is usually described in terms of the map $f\omega \mapsto (Res(P_j, f\omega) : 1 \leq j \leq n)$, so that the *Residue theorem*:

$$\sum_P Res(P, hf\omega) = 0$$

can be invoked to show the duality of the two codes, with residue having the standard meaning as the coefficient of $1/t(P)$ in the Laurent series expansion. (In the literature all that is usually found is the use of entries $f(P)$ to define a generator matrix G or parity-check matrix H, with very little reference to the other, defined by evaluating $Res(P, f\omega)$. This is probably because those using functional encoding to get G don't usually use H, and those using it to define H, limit their investigations to decoding, ignoring G.)

Riemann-Roch can also be used to show that $dim(\mathcal{L}(\mathcal{G})) \geq deg(\mathcal{G}) - (g - 1)$ with equality when $deg(\mathcal{G}) > deg(\omega)$. And it can be used as well to show that $d \geq (n-k) - (g-1)$, explaining g as a penalty relative to the Singleton bound alluded to earlier.

3 Weighted total-degree orders

There are many good books covering introductory material on Gröbner bases, with Cox, Little, and O'Shea [2] being the author's favorite. Much information is in the Gröbner Technology section of this volume, written by Teo Mora; but the emphasis there is not on the orderings (called *term orders* there). So the following is an addendum to said section for this purpose. The (default) *lexicogrphical order* is based on comparing products by considering their *indices* (that is, *exponents*) lexicographically. So, for instance, in the multivariate polynomial ring $R[x_3, x_2, x_1]$ the order looks like

$$1 \prec x_1 \prec x_1^2 \prec \cdots \prec x_2 \prec x_2x_1 \prec \cdots \prec x_2^2 \prec \cdots \prec x_3 \prec \cdots$$

which can be described by $x_3^{i_3}x_2^{i_2}x_1^{i_1} \succ x_3^{j_3}x_2^{j_2}x_1^{j_1}$ iff $(i_3 > j_3)$ or $(i_3 = j_3$ and $i_2 > j_2)$ or $(i_3 = j_3$ and $i_2 = j_2$ and $i_1 > j_1)$. The *total-degree orders* of interest here are the *grevlex* and *wtdeg* orders (short for *graded reverse lexicographical* and *weighted total degree*), in which (weighted) total degree is the first concern.

These can be defined in ways similar to that above; but the non-singular matrices

$$A_{grevlex} := \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \; A_{wtdeg} := \begin{pmatrix} w_3 & w_2 & w_1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

can be used to reduce these to the lexicographical case by converting the column vector of exponents $\begin{pmatrix} i_3 \\ i_2 \\ i_1 \end{pmatrix}$ to $\begin{pmatrix} i_3 + i_2 + i_1 \\ i_3 + i_2 \\ i_3 \end{pmatrix}$ or $\begin{pmatrix} w_3i_3 + w_2i_2 + w_1i_1 \\ i_3 + i_2 \\ i_3 \end{pmatrix}$ respectively.

4 Hermitian codes and affine-variety codes

The most common (and easily implemented) examples of *one-point AG codes* are those from *Hemritian curves* alluded to above; but, at the same time, these least exemplify the general theory, as shall be described here. Let

$q := p^m$, p a prime. Then the affine equation $Trace(y) := y^q + y = x^{q+1} =: Norm(x)$ defines an Hermitian curve in characteristic p by

$$\mathcal{X} := \{(x_2 : x_1 : x_0) \in \mathbf{P}^2(\overline{F}_p) \ : \ x_2^q x_0 + x_2 x_0^q - x_1^{q+1} = 0\}$$

The rational functions $f_q := x_1/x_0$ and $f_{q+1} := x_2/x_0$ have divisors

$$(f_q) = (-q) \cdot P_\infty + \sum_{i=0}^{q-1} 1 \cdot P_i, \quad (f_{q+1}) = (-q-1) \cdot P_\infty + (q+1) \cdot P_0$$

with $P_\infty := (1 : 0 : 0)$ and $P_i := (\alpha_i : 0 : 1)$, $\alpha_0 := 0$, and $\alpha_i^{q-1} + 1 = 0$, $1 \leq i \leq q-1$ defined over $\mathbf{F}_{q^2}$ if p is odd, $\mathbf{F}_q$ if p is even.

These curves are already in one-point form, since f_q and f_{q+1} clearly have no poles except at P_∞. Also the *differential* $\omega := d(f_q)$ has divisor $(\omega) = (q-2)(q+1) \cdot P_\infty$. So here $\omega = 1d(t_P)$ for all $P \neq P_\infty$. And the genus $g = q(q-1)/2$, meaning the (rational) functions $f_{(q+1)i+qj} := f_{q+1}^i f_q^j$, $0 \leq i < q$, $0 \leq j$ can be used to index the rows of both generator and parity-check matrices for the corresponding linear codes, normally chosen over $\mathbf{F}_{q^2}$, since there are $1 + q^3 = q^2 + 1 + 2g\sqrt{q^2}$ rational points there.

If $\mathbf{I}_{q^2} := \langle f_{q+1}^{q^2} - f_{q+1},\ f_q^{q^2} - f_q\rangle$, then $\mathbf{I} := \langle \mathbf{I}_{q^2},\ f_{q+1}^q - f_q^{q+1} + f_{q+1}\rangle$ has Gröbner basis $f_{q+1}^q - f_q^{q+1} + f_{q+1},\ f_q^{q^2} - f_q\rangle$, with

$$\Delta(I) = \{f_{q+1}^i f_q^j,\ 0 \leq i < q,\ 0 \leq j < q^2\}$$

of size $n = q^3$. In the examples below, with $q = 4$, the elements of $\{0, \gamma^0, \gamma^5, \gamma^{10}\}$ have trace 0, those of $\{\gamma^1, \gamma^2, \gamma^4, \gamma^8\}$ have trace 1, those of $\{\gamma^6, \gamma^9, \gamma^7, \gamma^{13}\}$ have trace γ^5, and those of $\{\gamma^{12}, \gamma^3, \gamma^{14}, \gamma^{11}\}$ have trace γ^{10}. And $x^i = 1, \gamma^5, \gamma^{10}$ for $i \equiv 0, 1, 2 (mod 3)$ respectively. This describes all 64 points other than P_∞.

Some of the reasons why these curves are atypical are:

1. most good curves are not given initially in one-point form;
2. most curves (in affine form) are not described by only one defining relation in two variables;
3. there are usually functions (used to define either the generator or parity-check matrices) that are not monomials in the given variables;
4. most codes need different sets of functions to define generator and parity-check matrices, because (ω) is not usually a multiple of P_∞.

It is also possible to think of AG codes strictly in terms of ideals in multivariate polynomial rings, meaning it is possible to choose a (necessarily finite) set of affine points $V \subseteq (\mathbf{F}_q)^s$, produce by interpolation the ideal $\mathbf{I}(V)$ having V as its variety, and think of $\mathcal{X}$ as having intersection V with

$(\mathbf{F}_q)^s$, without ever really defining it. The codes gotten by evaluating some $f \in \Delta(I)$ on the points of V are called *affine-variety codes* for obvious reasons. *Reed-Muller codes* (and hence *extended Reed-Solomon codes* as well) can be viewed this way with $V := \mathbf{F}_q^s$ and $\mathbf{I} := \mathbf{I}_q := \langle x_1^q - x_1, \dots, x_s^q - x_s \rangle$. Conceivably it is possible to consider all these as AG codes, but it is probably better, in general, to limit the usage of the term *AG code* to $\mathbf{V} \subseteq \mathbf{V}(I)$ for I describing a curve (or surface), since any linear code over $\mathbf{F}_q$ can be viewed as an affine-variety code.

5 Curve definition

Producing descriptions of codes from curves can be done in various ways. For instance, the standard projective description of the *Klein quartic* in characteristic 2 (which has 24 rational points over $\mathbf{F}_8$) is $x_2^3x_0+x_1^3x_2+x_0^3x_1 = 0$, but a proper description for use as a *one-point AG code* is given in affine terms by $f_7^2 + f_5f_3^3 + f_7 = 0$, $f_7f_5 + f_3^4 + f_5 = 0$, and $f_5^2 + f_7f_3 = 0$ for $f_3 := x_2/x_0$, $f_5 := x_2x_1/x_0^2$, and $f_7 := x_2x_1^2/x_0^3$, this being an induced Gröbner basis gotten from the standard description after changing variables. (21 of the $24-1$ affine points over $\mathbf{F}_8 := \mathbf{F}_2[\beta]/(1+\beta+\beta^3)$ have $f_3(P) := \beta^{5i}$, $f_5(P) := \beta^{6i}\delta_j$ and $f_7(P) := \delta_j^2$ for $\delta_j^3 + \delta_j + 1 = 0$; the other two having $f_3(P) = 0 = f_5(P)$ and $f_7(P) \in \{0, 1\}$.) And there is a two-point description given by $x_1^3x_2 + x_2^3(x_2 + x_1 + x_0) + (x_2 + x_1 + x_0)^3x_1 = 0$ with the 2 points at infinity, rational over $\mathbf{F}_4$, not $\mathbf{F}_8$.)

One-point AG codes, that is, those described by the functions from $\mathcal{L}(m \cdot P_\infty)$ (with poles only at P_∞ and of pole order at most m there), will have an $\overline{\mathbf{F}}_q[f_\rho]$-module basis of size ρ with elements having smallest possible pole sizes for each $i\ mod(\rho)$; said functions describing the multivariate polynomial ring used, and an ideal of relations describing the multiplication of those functions, modulo $\mathcal{X}$.

The problem is that $\mathcal{X}$ is not usually given in a such a friendly manner. Consider the curve described by:

$$x_2^2 + x_2 = \frac{x_1^2}{x_1 + 1},\ x_3^2 + x_3 = \frac{x_2^2}{x_2 + 1}$$

in characteristic 2, an example from the second towers found by Garcia and Stichtenoth. While this may define the curve $\mathcal{X}$, it doesn't allow for evaluation at several points at which some of the variables have poles. (In fact some points P cannot even be described in terms of the projective coordinates $x_3(P)$, $x_2(P)$, $x_1(P)$, and $x_0(P)$.)

The following MAGMA code first uses a change of variables to x_4, $x_6 := x_2(x_4 + 1)$ and $x_7 := x_1(x_2 + 1)(x_4 + 1)$ with poles of orders 4, 6, and 7 at some point P_∞ and no other poles, followed by a computation of a value $\Delta \in \mathbf{F}_2[x_4]$ such that $\mathcal{L}(m \cdot P_\infty) \subseteq 1/\Delta\overline{\mathbf{F}}_2\langle 1, y_6, y_7, y_7y_6 \rangle/\mathcal{I}$ for $\mathcal{I} :=$

$\langle x_7^2 + x_7(x_6 + x_4) + x_6x_4^2,\ x_6^2 + x_6(x_4 + 1) + x_4^2(x_4 + 1)\rangle$, followed by an implementation of the author's *q-th power algorithm* (that suffices for this and some similar examples) to produce the *missing function* y_5, to get a full set of parity-check functions $\mathcal{L}(m \cdot P_\infty$ for the AG codes related to this curve.

```
q:=2;
F:=FiniteField(q);
wt:=[7,6,4];
n:=#wt;r:=n-1;
P<x1,x2,x7,x6,x4>:=PolynomialRing(F,n+r);
e1:=x1*(x1+1)*(x2+1)-x2^2;
e2:=x2*(x2+1)*(x4+1)-x4^2;
def6:=x2*(x4+1)-x6;
def7:=x1*(x2+1)*(x4+1)-x7;
I:=ideal<P|e1,e2,def6,def7>;
EI:=EliminationIdeal(I,r);
GI:=GroebnerBasis(EI);
JM:=JacobianMatrix(GI);
M:=Minors(JM,n-1);
J:=ideal<P|GI,M>;
GJ:=GroebnerBasis(J);
d:=GJ[#GJ];
"---------------------------------";
R:=PolynomialRing(F,n,"grevlexw",wt);
AssignNames(~R,["y7","y6","y4"]);
x:=R.n;
hPR:=hom<P->R|0,0,R.1,R.2,R.3>;
wR:=function(ff)
   return &+[Degree(LeadingMonomial(ff),R.i)*wt[i]: i in [1..n]];
end function;
IR:=ideal<R|[GI[i]@hPR:i in [1..#GI]]>;
GR:=GroebnerBasis(IR);
QR:=quo<R|IR,R.n-1>;
NO:=MonomialBasis(QR);
h0:=hom<QR->R|[R.i:i in [1..n]]>;
MO:=h0(NO);
dR:=d@hPR;"Delta=",dR;
"---------------------------------";
for i in [1..#MO] do
   "B[",wR(MO[i])-wR(dR),"]=",MO[i];
end for;
DR:=(dR)^(q-1);
nf:=function(gg)
   return NormalForm(gg^q,IR);
end function;
LT:=function(ff) return LeadingTerm(ff); end function;
LM:=function(ff) return LeadingMonomial(ff); end function;
LC:=function(ff) return LeadingCoefficient(ff); end function;
N:=wR(dR)+1+Maximum([wR(MO[i]):i in [1..#MO]]);
zero:=[R|0: i in [1..N]];

change:=true;
while change do
   change:=false;
   g_new:=zero;
   k:=zero;
   Bnew:=[];
   for i in [1..#MO] do
      j:=1+wR(MO[i]);
      g_new[j]:=MO[i];
      k[j]:=nf(g_new[j]);
   end for;
"------------------------------";
   loop:=true;
   i:=1;
   while loop and (i le N) do
      loop:=false;
      empty:=false;
      if g_new[i] eq 0 then
         loop:=true;
         i+:=1;
         empty:=true;
      end if;
      if empty eq false then
         if k[i] eq 0 then
            Append(~Bnew,i);
            "B[",i-wR(dR)-1,"]=",g_new[i];
            loop:=true;
            i+:=1;
         else
            for j in [1..#M0] do
               bool,qu:=IsDivisibleBy(LT(k[i]),LT(M0[j]*DR));
               if bool then
                  k[i]-:=qu*M0[j]*DR;
                  loop:=true;
                  break;
               end if;
            end for;
            if (i gt 1) and (loop eq false) then
               lki:=LM(k[i]);
               for j in [1..i-1] do
                  if k[j] ne 0 then
                     if lki eq LM(k[j]) then
                        lc:=LC(k[i])/LC(k[j]);
                        g_new[i]-:=lc*g_new[j];
                        k[i]-:=lc*k[j];
                        loop:=true;
                        change:=true;
                        break;
                     end if;
                  end if;
               end for;
            end if;
            if loop eq false then
               j:=i+wt[n];
               g_new[j]:=g_new[i]*x;
               k[j]:=k[i]*x^q;
               loop:=true;
               change:=true;
               i+:=1;
            end if;
         end if;
      end if;
   end while;
   M0:=[g_new[i]:i in Bnew];
end while;

W:=wR(dR)+1;
"------------------------------------------";
S:=PolynomialRing(F,4,"grevlexw",[7,6,5,4]);
AssignNames(~S,["s7","s6","s5","s4"]);
hSR:=hom<S->R|g_new[7+W],g_new[6+W],g_new[5+W],g_new[W]*x>;
nn:=wt[n];
hRS:=hom<R->S|0,0,S.nn>;
BB:=[R|0: i in [1..nn]];
for i in Bnew do
   j:=wR(LT(g_new[i])) mod nn +1;
   BB[j]:=g_new[i];
end for;
s:=[S.(nn+1-i):i in [1..nn]];
s[1]:=1;
module:=function(i,j)

   ff:=NormalForm((s[i]*s[j])@hSR,IR) div dR;
   MM:=[R|0: i in [1..nn]];
   while ff ne 0 do
      j:=wR(LM(ff)) mod nn +1;
      k:=LT(ff) div LT(BB[j]);
      ff-:=k*BB[j];
      MM[j]+:=k;
   end while;
   return MM;
end function;
SS:=[S|];
for i in [2..nn] do
   for j in [2..i] do
      mm:=module(i,j)@hRS;
      ms:= &+[mm[k]*s[k]:k in [1..nn]];
      nij:=(s[i]*s[j])-ms;
      Append(~SS,nij);
   end for;
end for;
SI:=ideal<S|SS>;
SG:=GroebnerBasis(SI);
"A Grobner basis for the induced ideal is",SG;
```

Since there are other methods of producing the these functions (such as using MAGMA's *IntegralClosure* function or SINGULAR's *normal* function), perhaps a few words of explanation are in order, though there are various descriptions available in [5], [6], and [7]. The *integral closure* $ic(R)$ of a ring R (in this case, $R = \mathbf{F}_2[x_7, x_6, x_4]/\mathcal{I}$) in its *field of fractions* is the *ring* of all elements satisfying a monic (that is, having leading coefficient 1) polynomial with coefficients from R (in this case, $ic(R) = \cup_m \mathcal{L}(m \cdot P_\infty)$). Standard methods produce $ic(R)$ by finding a nested sequence of larger and larger rings living between R and $ic(R)$. The *q-th power algorithm* finds a module M_0 of the form $\Delta^{-1}R$, (with Δ only involving the independent variable(s), in this case just x_4) known to contain $ic(R)$, and then produces a nested sequence of smaller modules known to stabilize at $ic(R)$. These modules are easily described mathematically by $M_{i+1} := \{f \in M_i : NormalForm(f^q, \mathcal{I}) \in M_i\}$, and easily produced by an algorithm (such as the single-variable implementaion given above) which is *linear* over $\mathbf{F}_q$. (The competing algorithms were designed to work over characteristic 0, to work on number fields, and not with any idea of weights in mind; so they do not immediately provide an appropriate description for this type of application.)

The output of the MAGMA program above, modified to be more readable, gives $\Delta := y_4^2$, and $\mathbf{F}_2$-module bases for ΔM_i as $B_0 := \{1, y_6, y_7, y_7y_6\}$, $B_1 := \{y_4, y_6y_4, y_7y_4, y_7y_6\}$, $B_2 := \{y_4^2, y_7y_6 + y_6y_4, y_6y_4^2, y_7y_4^2\} =: B_3$, meaning the missing function is $y_5 := (y_7y_6 + y_6y_4)/y_4^2$. The curve can then be properly defined in *one-point form* as $\overline{\mathbf{F}}_2[y_7, y_6, y_5, y_4]/\mathcal{J}$ for $\mathcal{J}$ the ideal of induced relations (also produced by the code above) as

$$\mathcal{J} := \langle \begin{array}{l} y_7^2 + y_6y_4^2 + y_5y_4^2 + y_7y_4 + y_6y_4 + y_7, \\ y_7y_6 + y_5y_4^2 + y_6y_4, \\ y_6^2 + y_4^3 + y_6y_4 + y_4^2 + y_6, \\ y_7y_5 + y_4^3 + y_7y_4 + y_6y_4 + y_5y_4 + y_4^2 + y_7, \\ y_6y_5 + y_7y_4 + y_5y_4 + y_4^2 + y_7 + y_5 + y_4, \\ y_5^2 + y_6y_4 + y_5y_4 + y_4^2 + y_6 + y_5 + y_4 \rangle. \end{array}.$$

The codes over $\mathbf{F}_4$ are then functionally encoded or functionally decoded relative to (some of) the $n = 13$ functions $f_0 := 1, f_4 := y_4, f_5 := y_5, f_6 := y_6, f_7 := y_7, f_8 := y_4^2, f_9 := y_5y_4, f_{10} := y_6y_4, f_{11} := y_7y_4, f_{12} := y_4^3, f_{13} := y_5y_4^2, f_{14} := y_6y_4^2, f_{15} := y_7y_4^2$ as generator or parity-check functions respectively. The $n = 13$ affine points rational over $\mathbf{F}_4$ correspond to the *variety* of the above ideal $\mathcal{J}$ (intersected with $\langle y_4^4 - y_4, y_5^4 - y_5, y_6^4 - y_6, y_7^4 - y_7\rangle$ to restrict the curve to that subfield. In particular $(y_7(P), y_6(P), y_5(P), y_4(P))$ is a coordinatization of the affine point P.

A slightly more complicated example of producing a one-point AG code description is :

$$p_I(T) := \sum_{k=0}^{m} a_k T^k \in \mathbf{F}_q[T]$$

be monic and irreducible. Define inductively, $P_k \in \mathbf{F}_q[x, y]$ by $P_0(x, y) := 1$ and

$$P_{k+1}(x, y) := xP_k(x, y)^{q^2} + yP_k(x, y)^q \quad \text{for } k \geq 0.$$

Then let

$$F(x,y) = F_I(x,y) := \sum_{k=0}^{m} a_k P_k(x,y).$$

The equation $F_I(x,y)=0$ is an analogue of the modular equation.

As a small example, let $q := 2$ and $p_I(T) := 1 + T + T^3$. Then

$$P_1(x,y) = x + y, \quad P_2(x,y) = x^5 + x^2y + xy^4 + y^3,$$

$$P_3(x,y) = x^{21} + x^{10}y + x^9y^4 + x^5y^{16} + x^4y^3 + x^2y^9 + xy^{12} + y^7,$$

$$F(x,y) = x^{21} + x^{10}y + x^9y^4 + x^5y^{16} + x^4y^3 + x^2y^9 + x(y^{12}+1) + (y^7+y+1).$$

Start by using $x_1 := x$ and $x_2 := y + x$ to get

$$x_1^5(x_2^{16} + x_2^8) + x_1^3(x_2^8 + x_2^4) + x_1^2(x_2^9 + x_2^5) + x_1(x_2^{12} + x_2^6) + (x_2^7 + x_2 + 1),$$

$$(x_1) = (-8)\cdot P_1 + (-4)\cdot P_2 + (-4)\cdot P_3 + 5\cdot P_4 + 4\cdot P_5 + \sum_{j=1}^{7} 1\cdot Q_j,$$

$$(x_2) = 0\cdot P_1 + 2\cdot P_2 + 3\cdot P_3 + (-1)\cdot P_4 + (-4)\cdot P_5 + \sum_{j=1}^{7} 0\cdot Q_j.$$

Use $y_2 := x_1x_2$ to get

$$x_1^{11} + x_1^{10}(y_2^4 + y_2) + x_1^8(y_2^8 + y_2^5) + x_1^6(y_2^8 + y_2^6) + x_1^4(y_2^9 + y_2^7) + (y_2^{16} + y_2^{12}),$$

$$(y_2) = (-8)\cdot P_1 + (-2)\cdot P_2 + (-1)\cdot P_3 + 4\cdot P_4 + 0\cdot P_5 + \sum_{j=1}^{7} 1\cdot Q_j.$$

Use $y_1 := x_1/y_2$, to get

$$y_1^{11} + y_1^{10}(y_2^3 + 1) + y_1^8(y_2^5 + y_2^2) + y_1^6(y_2^3 + y_2) + y_1^4(y_2^2 + 1) + (y_2^5 + y_2),$$

$$(y_1) = 0\cdot P_1 + (-2)\cdot P_2 + (-3)\cdot P_3 + 1\cdot P_4 + 4\cdot P_5 + \sum_{j=1}^{7} 0\cdot Q_j.$$

Use $z_2 := y_2(y_1+1)^2$ to get

$$z_2^5 + z_2^3y_1^6 + z_2^2(y_1^6 + y_1^4) + z_2(y_1^6 + 1) + y_1^4(y_1+1)^2(y_1^7 + y_1 + 1),$$

$$(z_2) = 2\cdot P_1 + (-6)\cdot P_2 + (-7)\cdot P_3 + 4\cdot P_4 + 0\cdot P_5 + \sum_{j=1}^{7} 1\cdot Q_j.$$

Use

$$h_{21} := y_1^7 + z_2^2y_1 + z_2y_1^2 + z_2 + y_1^2,$$

$$h_{25} := z_2y_1^6 + z_2^3 + z_2^2y_1 + z_2y_1 + y_1^5 + y_1^3 + y_1^2 + 1,$$

to get a single defining relation:

$$\begin{gathered}
h_{25}^{21} + h_{25}^{20}h_{21} + h_{25}^{18}(h_{21}^{3} + h_{21} + 1) + h_{25}^{17}(h_{21}^{3} + 1) \\
+h_{25}^{16}(h_{21}^{4} + h_{21}) + h_{25}^{15}(h_{21}^{7} + h_{21}^{6} + h_{21}^{3} + h_{21} + 1) + h_{25}^{14}h_{21}^{7} \\
+h_{25}^{13}(h_{21}^{8} + h_{21}^{7} + h_{21}^{6} + h_{21}^{4} + h_{21}^{3} + 1) + h_{25}^{12}(h_{21}^{9} + h_{21}^{8} + h_{21}^{4} + 1) \\
+h_{25}^{11}(h_{21}^{11} + h_{21}^{9} + h_{21}^{8} + h_{21}^{5} + h_{21}^{4} + h_{21}^{3} + h_{21}^{2}) \\
+h_{25}^{10}(h_{21}^{12} + h_{21}^{9} + h_{21}^{8} + h_{21}^{7} + h_{21}^{5} + h_{21}^{3} + h_{21} + 1) \\
+h_{25}^{9}(h_{21}^{14} + h_{21}^{13} + h_{21}^{10} + h_{21}^{9} + h_{21}^{8} + h_{21}^{7} + h_{21}^{6} + h_{21}^{3} + h_{21}^{2} + 1) \\
+h_{25}^{8}(h_{21}^{13} + h_{21}^{9} + h_{21}^{8} + h_{21}^{6} + h_{21}^{4} + h_{21}^{3} + h_{21}) \\
+h_{25}^{7}(h_{21}^{16} + h_{21}^{15} + h_{21}^{13} + h_{21}^{12} + h_{21}^{11} + h_{21}^{7} + h_{21}^{3} + h_{21}) \\
+h_{25}^{6}(h_{21}^{17} + h_{21}^{16} + h_{21}^{13} + h_{21}^{9} + h_{21}^{8} + h_{21}) \\
+h_{25}^{5}(h_{21}^{17} + h_{25}^{16} + h_{25}^{12} + h_{21}^{7} + h_{21}^{5} + h_{21}^{2} + h_{21} + 1) \\
+h_{25}^{4}(h_{21}^{19} + h_{21}^{16} + h_{21}^{15} + h_{21}^{12} + h_{21}^{6} + h_{21}^{5} + h_{21}^{3} + 1) \\
+h_{25}^{3}(h_{21}^{18} + h_{21}^{15} + h_{21}^{12} + h_{21}^{10} + h_{21}^{9} + h_{21}^{7} + h_{21}^{4} + h_{21}) \\
+h_{25}^{2}(h_{21}^{22} + h_{21}^{21} + h_{21}^{20} + h_{21}^{18} + h_{21}^{13} + h_{21}^{12} + h_{21}^{9} + h_{21}^{8} + h_{21}^{7} + h_{21}^{5} + h_{21}^{4} + h_{21}^{3}) \\
+h_{25}(h_{21}^{23} + h_{21}^{22} + h_{21}^{20} + h_{21}^{17} + h_{21}^{15} + h_{21}^{14} + h_{21}^{12} + h_{21}^{9}) \\
+(h_{21}^{25} + h_{21}^{23} + h_{21}^{19} + h_{21}^{17} + h_{21}^{15} + h_{21}^{13} + h_{21}^{11} + h_{21}^{5}).
\end{gathered}$$

But then the q-th power algorithm produces the integral closure:

$$\mathbf{F}_2[h_{16}, h_{15}, h_{13}, h_{12}, h_{11}, h_{10}, h_7]/\mathcal{I};$$

with $\mathcal{I}$ having Gröbner basis consisting of

$$\begin{gathered}
h_{10}^{2} + h_{13}h_7 + h_{11}h_7 + h_{10}h_7 + h_{15} + h_{13}, \\
h_{11}h_{10} + h_7^{3} + h_{13}h_7 + h_{15} + h_{13} + h_{11} + h_{10}, \\
h_{11}^{2} + h_{15}h_7 + h_{13}h_7 + h_{12}h_7 + h_{15} + h_{13} + h_{11} + h_{10}, \\
h_{12}h_{10} + h_{15}h_7 + h_{13}h_7 + h_{12}h_7 + h_{12}, \\
h_{12}h_{11} + h_{16}h_7 + h_{12}h_7 + h_{10}h_7, \\
h_{12}^{2} + h_{10}h_7^{2} + h_{16}h_7 + h_{15}h_7 + h_7^{3} + h_{13}h_7 + h_{12}h_7 + h_{11}h_7 + h_{14} + h_{12} + h_7, \\
h_{13}h_{10} + h_{16}h_7 + h_{15}h_7 + h_{13}h_7 + h_{12}h_7 + h_{11}h_7 + h_7^{2} + h_{12} + h_7, \\
h_{13}h_{11} + h_{10}h_7^{2} + h_{16}h_7 + h_{11}h_7 + h_{13} + h_{10} + h_7 + 1, \\
h_{13}h_{12} + h_{11}h_7^{2} + h_{15}h_7 + h_{12}, \\
h_{13}^{2} + h_{12}h_7^{2} + h_{10}h_7^{2} + h_{15}h_7 + h_{15}h_7^{2} + 1, \\
h_{15}h_{10} + h_{11}h_7^{2} + h_{13}h_7 + h_{15} + h_7, \\
h_{15}h_{11} + h_{12}h_7^{2} + h_{11}h_7^{2} + h_{10}h_7^{2} + h_{15}h_7 + h_{13}h_7 + h_{15} + h_7^{2} + h_7, \\
h_{15}h_{12} + h_{13}h_7^{2} + h_{12}h_7^{2} + h_{10}h_7^{2} + h_{15}h_7, \\
h_{15}h_{13} + h_7^{4} + h_{13}h_7^{2} + h_{12}h_7^{2} + h_{10}h_7^{2} + h_{15}h_7 + h_{10}h_7 + h_7^{2} + h_7, \\
h_{15}^{2} + h_{16}h_7^{2} + h_7^{4} + h_{12}h_7^{2} + h_{11}h_7^{2} + h_{10}h_7^{2} + h_{15}h_7 + h_{10}h_7 + h_{15} + h_7^{2} + h_7, \\
h_{16}h_{10} + h_{12}h_7^{2} + h_{11}h_7^{2} + h_{13}h_7 + h_{16} + h_{10} + h_7 + 1, \\
h_{16}h_{11} + h_{13}h_7^{2} + h_{11}h_7^{2} + h_{15}h_7 + h_{13}h_7 + h_{12}h_7 + h_{16} + h_7^{2} + h_{11} + h_7 + 1, \\
h_{16}h_{12} + h_7^{4} + h_{12}h_7^{2} + h_{11}h_7^{2} + h_{10}h_7^{2} + h_{16}h_7 + h_{12}h_7 + h_{12} + h_7,
\end{gathered}$$

$$h_{16}h_{13} + h_7^4 + h_{13}h_7^2 + h_{10}h_7^2 + h_{16}h_7 + h_{10}h_7 + h_{16} + h_{15} + h_7^2 + h_{13} + 1,$$

$$h_{16}h_{15} + h_{10}h_7^3 + h_7^4 + h_{12}h_7^2 + h_{11}h_7^2 + h_{10}h_7^2 + h_{15}h_7 + h_7^3 + h_{10}h_7 + h_7,$$

$$h_{16}^2 + h_{11}h_7^3 + h_{16}h_7^2 + h_{15}h_7^2 + h_7^4 + h_{13}h_7^2 + h_{15}h_7 + h_{12}h_7 + h_{10}h_7 + h_{16} + h_7^2 + h_7.$$

As a by-product, the smallest type I representation (relative to this choice of P_∞) would then be in terms of the single polynomial relating h_{10} and h_7:

$$h_{10}^7 + h_{10}^6h_7 + h_{10}^5(h_7^2 + 1) + h_{10}^4(h_7 + 1) + h_{10}^3(h_7^5 + h_7^4 + h_7^2 + 1) + h_{10}^2(h_7^7 + h_7^6 + h_7^3 + h_7^2 + 1)$$

$$+h_{10}(h_7^7 + h_7^6 + h_7^5 + 1) + (h_7^{10} + h_7^9 + h_7^8 + h_7^7 + h_7^4 + h_7^3 + h_7^2 + h_7 + 1).$$

References

[1] P. Beelen, A. Garcia, and H. Stichtenoth, "Towards a classification of recursive towers of function fields over finite fields", Finite Fields and Their Applications, vol. 12, Jan. 2006, pp. 56-77.

[2] D. Cox, J. Little, and D. O'Shea, Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra, 3rd Edition Springer, New York, 2007.

[3] A. Garcia and H. Stichtenoth "On the asymptotic behaviour of some towers of function fields over finite fields", J. Number Theory, vol. 61, pp. 248-273, 1996.

[4] A. Garcia and H. Stichtenoth, "Asymptotics for the genus and the number of rational places in towers of function fields over finite fields", Finite Fields and Their Applications, vol. 11, pp. 434-450, August, 2005.

[5] "Finding the Defining Functions of One-Point Algebraic-Geometry Codes", IEEE Trans. Inform. Theory, vol. 47, pp. 2566-2573, Sept., 2001.

[6] D. Leonard and R. Pellikaan "Integral closures and weight functions over finite fields" Finite Fields and Their Applications, vol. 9, pp. 479-504, Oct. 2003.

[7] D. Leonard, "A module view of integral closures", submitted.

[8] The MAGMA Computational Algebra System for Algebra, Number Theory and Geometry. The University of Sydney Computational Algebra Group. `http://magma.maths.usyd.edu.au/magma`

[9] T. Høholdt, R. Pellikaan, and J. H. van Lint, Algebraic Geometry Codes, Handbook of Coding Theory, chapter 10, V. PLess and C. Huffman, eds., North Holland, Amsterdam, pp. 871-961, 1998.