

This domain defines a ring of differential operators which act on an A-module, where A is a differential ring. The type of the coefficients, A, is declared to belong to the category DifferentialRing and type of the operands, M, is declared to belong to the category Module(A) with a derivative operation. The constructed domain of operators is declared to belong to a category of general polynomials with coefficients A and two additional operations. The operation D creates a differential operator and "." provides the method of applying operators to elements of M.

Multiple Views and Multiple Inheritance

It is often necessary to view a given domain as belonging to different categories at different times. Sometimes we want to think of the domain Integer as belonging to Ring, sometimes as belonging to OrderedSet, and at other times as belonging to other categories. For a domain to have multiple views, it should be declared to belong to the Join of the appropriate categories. For example, the following keyed access file datatype may be viewed either as a table or as a file:

```
KeyedAccessFile(Entry: Set): T == B where
  FileRec ==> Record(key: String, entry: Entry)
  ErrorMsg ==> String

T == Join(FileCategory(LibraryName, FileRec),
          TableCategory(String, Entry, ErrorMsg))
with
  pack: $ -> $

B == add ...
```

An important use of categories is to supply default implementations of operations. So long as certain primitive operations are provided by a domain, others can be implemented categorically. For example, supplying only "<" and "=" allows definitions of ">", "<=" and ">=". Thus a domain may inherit operations from a category. The use of Join provides multiple inheritance.

Compile-Time Binding

A domain object contains several vectors of function/environment pairs. When operation bindings are not known at compile-time (as for the operations exported by a type parameter), the operations are performed by calling the function in the appropriate slot of a vector. When operation bindings are known at compile time, a code fragment for the operation may be placed in line.

The scope rules in Scratchpad II and the operations applicable to domain values have been designed so that when a domain is known to belong to a particular category, it is also known exactly what operations it exports. From this, the precise location of each function is determined. Thus when a function is called using the general mechanism, a hard-coded offset is used.

Scratchpad II is implemented on top of LISP/VM. Since the exact number and type of arguments are known at compile time, much of the usual function can be omitted. This, in conjunction with compile-time knowledge of function offsets, makes function calling in Scratchpad II faster than that of the underlying Lisp system.

Construction of Algebraic Error Control Codes (ECC) on the Elliptic Riemann Surface

by
Martin Hassner
William H. Burge
Stephen M. Watt

In this paper we make use of the power series facility of Scratchpad II to construct algebraic ECC on the elliptic Riemann surface of genus one. We present the construction of a specific example that highlights the method.

Linear algebraic ECC are formally defined as ideals in a function field. A message word which consists of k symbols in a ground field is mapped by a matrix encoder over the ground field into a code word which consists of n symbols, $n > k$. The encoder matrix is the null space of a check matrix that constrains the symbols of the code word to be the residues at n distinct singular points of first order of a function whose poles are a fixed set of pointers that locate the symbols inside each code word. An additional constraint imposed on each code function is its divisibility by a fixed function or linear system of functions whose zero set is disjoint from the pole set. A code consists of all the n -vectors of residues associated with such a system of functions which form an ideal. Within this algebraic framework it is possible to classify and determine (lower) bounds on the performance, i.e. efficiency vs. correction power, of algebraic ECC.

	P ₀	P ₁	P ₂	P ₃	P ₄	P ₅	P ₆	P ₇	P ₈
X ₀	1	0	α	1	0	β	0	α	1
X ₁	0	1	0	1	α	0	β	1	α
X ₂	1	1	1	0	1	1	1	0	0

Figure 1. 9 rational points

The bulk of the existing theory of algebraic ECC is centered on ideals of rational functions in the rational function field obtained as a simple transcendental extension of the ground field $GF(2^n)$. The choice of this ground field was dictated by applications of the theory to computer data. The theory was extended recently by the Russian coding theorist Goppa, who formulated the unifying algebraic framework presented above, to fields of algebraic functions in one variable obtained by algebraic extensions of the rational function field [2]. Further interest in this theoretical development was produced by a recent paper in which it was shown that codes defined on elliptic modular curves, i.e. in elliptic modular function fields, have asymptotic parameters that exceed the Gilbert bound, a lower bound on the asymptotic performance of linear codes [4]. This bound was believed to be tight until the appearance of this result.

The construction described in [2] uses a geometric model for the field of algebraic functions defined on an algebraic curve. As a specific example a code constructed on the nonsingular elliptic cubic over the ground field $GF(2^2) = \{0, 1, \alpha, \beta\}$ is presented. The equation of the nonsingular cubic over this ground field in the projective coordinates $\{X_0, X_1, X_2\}$ is

$$X_0^3 + X_1^3 + X_2^3 = 0 \tag{1}$$

On this curve there are 9 rational points with coordinates in this ground field (see Figure 1).

These 9 points are the inflection points of the cubic, each has a triple tangent. They constitute the pole divisor $D = \{P_i\}, i = 0, \dots, 8$, whose degree is the block length of each code word, $n = 9$. The linear system of functions chosen for the zero divisor G is $\{X_0^2, X_1^2, X_2^2, X_0X_1, X_1X_2, X_0X_2\} = \{\phi_i\}, i = 1, \dots, 6$. This linear system generates all the conics, in particular the conic $\phi_0 = X_0X_1 + X_1X_2 + X_0X_2$ which assumes nonzero values at all the nine points $\{P_i\}, i = 0, \dots, 8$. The check matrix is obtained as

$$\left\{ \frac{\phi_i}{\phi_0}(P_j) \right\}$$

and the generator matrix given in [2] is the following

$$\begin{pmatrix} 1 & 0 & 0 & \alpha & \alpha & \beta & 0 & \beta & 1 \\ 0 & 1 & 0 & \alpha & \beta & 0 & 1 & \beta & \alpha \\ 0 & 0 & 1 & 0 & 1 & \alpha & 1 & 1 & \alpha \end{pmatrix}$$

The number of points at which a conic intersects a cubic is 6, by Bezout's theorem, hence the degree of the zero divisor G is 6. The genus of the elliptic curve is $g = 1$. These two parameters determine the code separation d and the number of error control symbols r estimated through the Riemann-Roch theorem

$$H = \begin{pmatrix} 1 & 1 & 1 & q^{2/3} + 1 & \alpha q^{2/3} + \alpha & \beta q^{2/3} + \beta & \alpha q^{2/3} + \alpha & q^{2/3} + 1 & \beta q^{2/3} + \beta \\ q^{2/3} + 1 & \alpha q^{2/3} + \alpha & \beta q^{2/3} + \beta & q^{2/3} + 1 & \beta q^{2/3} + \beta & \alpha q^{2/3} + \alpha & 1 & 1 & 1 \\ q^{2/3} + 1 & \beta q^{2/3} + \beta & \alpha q^{2/3} + \alpha & 1 & 1 & 1 & \beta q^{2/3} + \beta & q^{2/3} + 1 & \alpha q^{2/3} + \alpha \\ q^{1/3} + 1 & \beta q^{1/3} + \beta & \alpha q^{1/3} + \alpha & q^{2/3} + 1 & q^{2/3} + 1 & q^{2/3} + 1 & \beta q^{1/3} + \beta & q^{1/3} + 1 & \alpha q^{1/3} + \alpha \\ q^{2/3} + 1 & q^{2/3} + 1 & q^{2/3} + 1 & q^{1/3} + 1 & \alpha q^{1/3} + \alpha & \beta q^{1/3} + \beta & \alpha q^{1/3} + \alpha & q^{1/3} + 1 & \beta q^{1/3} + \beta \\ q^{1/3} + 1 & \alpha q^{1/3} + \alpha & \beta q^{1/3} + \beta & q^{1/3} + 1 & \beta q^{1/3} + \beta & \alpha q^{1/3} + \alpha & \beta + 1 & \beta + 1 & \beta + 1 \end{pmatrix}$$

Figure 2. Parity check matrix of rank 6

$$\begin{aligned} d &\geq \text{deg}G - 2g + 2 \\ r &\leq d + g - 1 \end{aligned} \tag{2}$$

The actual parameters for the example considered are $d = 6, r = 6$.

We will now introduce an analytic model for the nonsingular elliptic cubic based on the Riemann surface for the elliptic normal curve $w = \sqrt{z(1-z)(1-k^2z)}$. This surface is a torus which becomes simply connected by introducing two cuts. It has two periods, the complete elliptic integrals of the first kind

$$K = \int_0^1 \frac{dz}{2w} \quad \text{and} \quad iK' = \int_1^{k^2} \frac{dz}{2w}$$

The elliptic integral of the first kind

$$u = \int_0^z \frac{dz}{2w}$$

maps the cut torus conformally onto a rectangle whose sizes are $2K$ and $2iK'$. The inverse map from this fundamental domain onto the torus determines the coordinates $\{z, w\}$ of a point u in the complex plane. This map is described by quotients of power series in the variable $q = e^{\pi i u / K}$ and named Theta-series in honor of Jacobi who used this notation. In particular the coordinates on the torus of points that lie on a regular grid inside the fundamental domain can be expressed in terms of quotients of Theta-series with broken characteristic [3]. For our particular example these are the 9 trisection points and the corresponding 9 characteristics are obtained by allowing g and h to assume values in the set $\{-1, 0, 1\}$; the argument v below is $u/2K$.

$$\vartheta \left[\begin{matrix} g \\ h \end{matrix} \right] (v) = \sum_{m=-\infty}^{m=\infty} q^{\binom{m+\frac{g}{3}}{3}} e^{2i\pi \binom{m+\frac{g}{3}}{3} \left(v + \frac{h\pi}{3} \right)} \tag{3}$$

The 9 trisection points are imbedded into the projective plane by triple products of these series such that the characteristic sum in each product is congruent to zero mod 3. The projective coordinates chosen are

$$\begin{aligned} X_0 &= \vartheta \left[\begin{matrix} 0 \\ 0 \end{matrix} \right] (v) \vartheta \left[\begin{matrix} 1 \\ 0 \end{matrix} \right] (v) \vartheta \left[\begin{matrix} -1 \\ 0 \end{matrix} \right] (v) \\ X_1 &= \vartheta \left[\begin{matrix} 0 \\ 1 \end{matrix} \right] (v) \vartheta \left[\begin{matrix} 1 \\ 1 \end{matrix} \right] (v) \vartheta \left[\begin{matrix} -1 \\ 1 \end{matrix} \right] (v) \\ X_2 &= \vartheta \left[\begin{matrix} -1 \\ -1 \end{matrix} \right] (v) \vartheta \left[\begin{matrix} 0 \\ -1 \end{matrix} \right] (v) \vartheta \left[\begin{matrix} 1 \\ -1 \end{matrix} \right] (v) \end{aligned} \tag{4}$$

The Sylvester form of the nonsingular cubic in the projective plane is a power series identity satisfied by these three power series

$$X_0^3 + X_1^3 + X_2^3 + 6m X_0 X_1 X_2 = 0 \tag{5}$$

The parameter m is a modular invariant that parametrizes the family of cubics. Its value is $m = -(1 + 2c^3)/6c^2$ where c is given in terms of theta-nulls [5]

$$c = \frac{\vartheta \left[\begin{matrix} 0 \\ 1 \end{matrix} \right] (0) \vartheta \left[\begin{matrix} 1 \\ 1 \end{matrix} \right] (0) \vartheta \left[\begin{matrix} -1 \\ 1 \end{matrix} \right] (0)}{\vartheta \left[\begin{matrix} 0 \\ 0 \end{matrix} \right] (0) \vartheta \left[\begin{matrix} 1 \\ 0 \end{matrix} \right] (0) \vartheta \left[\begin{matrix} -1 \\ 0 \end{matrix} \right] (0)} \tag{6}$$

The reason that equations (1) and (5) have different forms is that the coordinate origin in the projective plane in [2] is picked such that the term $6m X_0 X_1 X_2$ is zero. The coefficients of the power series in $X_0, X_1,$ and X_2 at the 9 trisection points are algebraic integers obtained from the cubic root of unity. In Scratchpad II by a simple coercion we force them to lie in $GF(2^3)$. The projective coordinates of all 9 trisection points are obtained by using the transformation formula for Theta-series[3]; the parameter τ in equation (7) is iK'/K .

$$\begin{aligned} \vartheta \left[\begin{matrix} g \\ h \end{matrix} \right] \left(v + g' \frac{\pi \tau}{3} + h' \frac{\pi}{3} \right) = \\ \vartheta \left[\begin{matrix} g + g' \\ h + h' \end{matrix} \right] (v) e^{-\frac{ig'}{3} \left(\frac{2\pi(h+h')}{3} + \frac{g'\pi\tau}{3} \right)} \end{aligned} \tag{7}$$

We have implemented in Scratchpad II the Jacobi Theta series in a package named THETA3 and with it we have checked equation (5). In the computation of the check matrix we make use of a Pade approximation algorithm [1] that was implemented in a Scratchpad II package named PADE which approximates $\{X_1/X_0, X_2/X_1\}$ by quotients of polynomials in $q^{1/3}$. We chose the lowest degree approximants with numerator of degree 1 and denominator of degree 2. The approximants satisfy equation (5) exactly. The parity check matrix is computed as $\{\phi_i(P_j)\}$ where we use the same linear system as in [2]. All the elements in this matrix are nonzero, it has rank 6, and it is given in Figure 2 on page 6.

The generator matrix obtained as the null space of this matrix is as follows

$$\begin{pmatrix} \alpha & 1 & \beta & \alpha & \beta & 1 & 0 & 0 & 0 \\ 0 & \alpha & \alpha & 1 & 1 & 0 & 1 & 1 & 0 \\ \alpha & 0 & \beta & \alpha & \alpha & 0 & 1 & 0 & 1 \end{pmatrix}$$

The code over $GF(2^2)$ generated by this matrix has distance $d=6$ and 6 control symbols, i.e., the same parameters as the code produced in [2]. Furthermore the weight enumerators of the two codes are identical, both have 63 nonzero code words of block length 9 out of which 36 have Hamming weight 6 and 27 have Hamming weight 8.

To conclude, we have shown an alternative method for the derivation of algebraic ECC on algebraic curves that seems to give equivalent codes. In comparison with the geometric model used by Goppa, the Riemann surface provides a local analytic structure which should be useful in the reconstruction of code functions, i.e. in decoding. So far this problem has remained unsolved within the framework of the geometric method.

The first author acknowledges the warm support of the Computer Algebra Group during his visit in the winter of 1987 as well as several stimulating discussions with Professors Chudnovsky from Columbia University.

References

- [1] Baker, G.A., and Graves-Morris, P.R., "Pade approximants: Basic Theory Part I", *Encyclopedia of Mathematics*, vol. 13, Reading, Massachusetts: Addison-Wesley, 1981.
- [2] Goppa, V.D., "Codes on algebraic curves," *Soviet Math.* 24 (1981): 170-172.
- [3] Krazer, A., *Lehrbuch der Thetafunktionen*, New York: Chelsea, 1970.
- [4] Katsman, G.L., Tsfasman, M.A. and Vladut, S.G., "Modular curves and codes with a polynomial construction," *IEEE Tr. on IT*, 30 (1984): 353-355.
- [5] Sievert, H., *Die Parameterdarstellung der Kurven 3. Ordnung durch Thetafunktionen*, Pr. Bayreuth, 1905.

Some Questions and Answers About Scratchpad II

The Computer Algebra Group welcomes your questions about Scratchpad II. Questions deemed to

be most interesting to a wide audience will be answered in future columns, while more specific questions will be answered on an individual basis.

Q: My terminal does not have brackets ("[" and "]"") or braces ("{" and "}")". Is there another way to enter these characters so that I can create lists and sets? prime number p dividing an integer q ?

A: The 2 character combinations ("[" and "]"") and (" $<$ " and " $>$ ") may be substituted for "[" and "]" and "{" and "}", respectively.

(1,2,3|)

(1) [1,2,3]

Type: L I

(<1,2,3>)

(2) {1,2,3}

Type: FSET I

Another way of accomplishing the same thing is to use the n-ary function *construct* instead of "[" and "]"". The function *brace* may be applied to a list to create a set.

construct(1,2,3)

(3) [1,2,3]

Type: L I

brace construct(1,2,3)

(4) {1,2,3}

Type: FSET I

Q: How do I evaluate a polynomial with a particular value for a variable?

A: Throughout Scratchpad II the function *eval* is used to do what you want. For example, given the polynomial

$p := x^{**3} - (x*y^{**2} - 2*z)^{**2}$

(1) $-4z^2 + 4x^2yz - x^2y^2 + x^3$

Type: P I

you can substitute 10 for x by issuing