The HERmitian Package

Divisors and Riemann-Roch Spaces of Algebraic Function Fields of Hermitian Curves

Version 0.1

14 March 2019

Gábor P. Nagy
Sabira El Khalfaoui

Gábor P. Nagy Email: nagyg@math.u-szeged.hu
Homepage: http://www.math.u-szeged.hu/~nagyg/

Sabira El Khalfaoui Email: sabira@math.u-szeged.hu

Copyright

Acknowledgements

We appreciate very much all past and future comments, suggestions and contributions to this package and its documentation provided by GAP users and developers.

Contents

Chapter 1

Introduction

This chapter describes the GAP package HERmitian. This package implements functionalities for divisors and Riemann-Roch spaces of an algebraic function field of Hermitian.

If you are viewing this with on-line help, type:

```
gap> ?HERmitian package
```

to see the functions provided by the HERmitian package.

1.1 Unpacking the HERmitian Package

If the HERmitian package was obtained as a part of the GAP distribution from the "Download" section of the GAP website, you may proceed to Section **??**. Alternatively, the HERmitian package may be installed using a separate archive, for example, for an update or an installation in a non-default location (see (**Reference: GAP Root Directories**)).

Below we describe the installation procedure for the `.tar.gz` archive format. Installation using other archive formats is performed in a similar way.

To install the HERmitian package, unpack the archive file, which should have a name of form `HERmitian-XXX.tar.gz` for some version number *XXX*, by typing

```
gzip -dc HERmitian-XXX.tar.gz | tar xpv
```

It may be unpacked in one of the following locations:

- in the `pkg` directory of your GAP 4 installation;
- or in a directory named `.gap/pkg` in your home directory (to be added to the GAP root directory unless GAP is started with `-r` option);
- or in a directory named `pkg` in another directory of your choice (e.g. in the directory `mygap` in your home directory).

In the latter case one one must start GAP with the `-l` option, e.g. if your private `pkg` directory is a subdirectory of `mygap` in your home directory you might type:

```
gap -l ";myhomedir/mygap"
```

where *myhomedir* is the path to your home directory, which (since GAP 4.3) may be replaced by a tilde (the empty path before the semicolon is filled in by the default path of the GAP 4 home directory).

1.2 Loading the HERmitian Package

To use the HERmitian Package you have to request it explicitly. This is done by calling `LoadPackage` (**Reference: LoadPackage**):

Example
```
gap> LoadPackage("HERmitian");
----------------------------------------------------------------
Loading  HERmitian 0.1
by Gábor P. Nagy (http://www.math.u-szeged.hu/~nagyg)
For help, type: ?HERmitian package
----------------------------------------------------------------
true
```

If GAP cannot find a working binary, the call to `LoadPackage` will still succeed but a warning is issued informing that the `HelloWorld()` function will be unavailable.

If you want to load the HERmitian package by default, you can put the `LoadPackage` command into your `gaprc` file (see Section (**Reference: The gap.ini and gaprc files**)).

1.3 Testing the HERmitian Package

You can run tests for the package by

Example
```
gap> Test(Filename(DirectoriesPackageLibrary("HERmitian"),"../tst/testall.tst"));
```

Chapter 2

Mathematical background

Our notation and terminology are standard. The reader is referred to [HKT08], [Sti09]. For the decoding of algebraic-geometric codes see the survey paper [HP95].

2.1 Algebraic curves, places, divisors

An algebraic plane curve X over the field K is given by a polynomial $f(X,Y) \in K[X,Y]$ of degree n; the usual notation is $X : f(X,Y) = 0$. The *affine points* of X are pairs $(x,y) \in L^2$, where L is an extension field of K and $f(x,y) = 0$ holds. We say that (x,y) is a *smooth point* of X if $(\frac{\partial f}{\partial X}(x,y), \frac{\partial f}{\partial Y}(x,y)) \neq (0,0)$. At a smooth affine point $(x,y) \in L^2$, the curve has formal local parametrization $(\xi(t), \eta(t)) \in L[[t]]^2$ such that $\xi(0) = x$, $\eta(0) = y$ and $f(\xi(t), \eta(t)) = 0$. Non smooth points are called *singular.*

The affine curve $X : f(X,Y) = 0$ has *homogeneous equation* $F(X,Y,Z) = 0$ with $F(X,Y,Z) = Z^n f(\frac{X}{Z}, \frac{Y}{Z})$. The *projective points* of X satisfy $F(x,y,z) = 0$. In particular, the affine point (x,y) of $\mathscr{X}$ corresponds to a projective point $(x : y : 1)$. The points of X at infinity are given by the homogeneous equation $F(X,Y,0) = 0$. Smoothness and local parametrization at projective points are defined in the obvious way. We say that the projective point $(x : y : z)$ of X is *defined over* L if $x/y, y/z, z/x$ are either infinite or in L. Notice that any singular point (affine or projective) is defined over an algebraic extension of the underlying field K.

The algebraic curve X is said to be *nonsingular* or *smooth,* if all its points are smooth. This implies that f is absolutely irreducible. For smooth algebraic plane curves, the concept of a *place* is equivalent with the concept of a point, when X is considered as a curve over the algebraic closure of K. A *divisor* is a formal sum $D = n_1P_1 + \ldots + n_kP_k$ with integers $n_1, \ldots, n_k$ and places $P_1, \ldots, P_k$. The degree of D is $n_1 + \cdots + n_k$. The integer n_i is the *valuation* $v_{P_i}(D)$ of D at P_i; for $P \neq P_i$ one has $v_P(D) = 0$. The *support* of D is the set of places P such that $v_P(D) \neq 0$.

2.2 Function fields and Riemann-Roch spaces

Let $X : f(X,Y) = 0$ be a smooth plane algebraic curve. The function field $K(X)$ of X is generated by the variables x,y subject to the algebraic relation $f(x,y) = 0$. In particular, each element of $K(X)$ can be written as $a(x,y)/b(x,y)$ with $a,b \in K[X,Y]$. Let $h \in K(X)$ and a place P of X, we define the valuation $v_P(h)$ as the subdegree of $h(\xi(t), \eta(t))$, where $(\xi(t), \eta(t))$ is the formal local parametrization at P. If $v_P(h) > 0$ then P is a *zero* of h, if $v_P(h) < 0$ then P is a *pole* of h. If $v_P(h) \geq 0$, then $h(P) = h(\xi(0), \eta(0))$ is a well-defined element of K.

For every non-zero function $h \in K(X)$, $\mathrm{Div}(h)$ stands for the principal divisor associated with h while $\mathrm{Div}(h)_0$ and $\mathrm{Div}(h)_\infty$ for its zero and pole divisor. Furthermore, for every separable function $h \in K(X)$, dh is the exact differential arising from h, and Ω denotes the set of all these differentials. Also, $\mathrm{res}_P(dh)$ is the residue of dh at a place of P of $K(X)$.

For any divisor A of $K(X)$, the *Riemann-Roch space* of A is

$$\mathscr{L}(A) = \{h \in K(X) \setminus \{0\} | \mathrm{Div}(h) \succeq -A\} \cup \{0\}.$$

We denote $\ell(A) = \dim(\mathscr{L}(A))$. Furthermore, the *differential space* of A is

$$\Omega(A) = \{dh \in \Omega \mid \mathrm{Div}(dh) \succeq A\} \cup \{0\}.$$

Both the Riemann-Roch and the differential spaces are linear spaces over K. Their dimensions are given by the theorem of Riemann-Roch:

$$\ell(A) = \deg(A) + 1 - g + \ell(W - A).$$

Here, W is a canonical divisor of X, and g is the *genus* of X. The latter is the most important birational invariant of an algebraic curve. For smooth curves of degree n, the genus formula is

$$g = \frac{(n-1)(n-2)}{2}.$$

The theorem of Riemann-Roch implies

$$\ell(A) \geq \deg(A) + 1 - g,$$

with equality if $\deg(A) > 2g - 2$.

2.3 Automorphisms of algebraic curves

Let $X : f(X,Y) = 0$ be a smooth plane algebraic curve with function field $K(X) = K(x,y)$, where the elements x, y are subject to the algebraic relation $f(x,y) = 0$. We assume that K is the constant field of $K(X)$. An *automorphism* of X is an automorphism of the function field, leaving all elements of K fixed. In particular, for any automorphism α of X, there are polynomials $u, v, w \in K[X,Y]$ such that

$$\alpha : (x,y) \to \left(\frac{u(x,y)}{w(x,y)}, \frac{v(x,y)}{w(x,y)}\right).$$

Substituting formal power series in α, we obtain an action of α on the set of places of X. This extends to an action on divisors, differentials and Riemann-Roch spaces.

2.4 Algebraic plane curves over finite fields

Let p be a prime and K an algebraically closed field of characteristic p. For $q = p^e$ we define the *Frobenius automorphism* $\mathrm{Frob}_q : x \mapsto x^q$ of K. This extends to an Frobenius map of K-polynomials (acting on the coefficients) and of affine and projective points over K (acting on the coordinates). The curve X is said to be $\mathbb{F}_q$-rational, if it is Frob_q-invariant. Moreover, the Frobenius action extends to places and divisors of $\mathbb{F}_q$-rational curves, which allows us to speak of places and divisors defined over

$\mathbb{F}_q$. Let X be an algebraic plane curve over $\mathbb{F}_q$ and P a place of X. Let r be the smallest positive integer such that P is defined over $\mathbb{F}_{q^r}$. Then, the divisor

$$P + P^{\mathrm{Frob}_q} + P^{\mathrm{Frob}_q^2} + \cdots + P^{\mathrm{Frob}_q^{r-1}}$$

is an $\mathbb{F}_q$-rational place of degree r of X.

If A is an $\mathbb{F}_q$-rational divisor then the Riemann-Roch space $\mathscr{L}(A)$ has a basis which consists of $\mathbb{F}_q$-rational elements of the function field of X. Hence, we can view $\mathscr{L}(A)$ as an $\mathbb{F}_q$-linear space of dimension $\ell(A)$. Similarly, $\Omega(A)$ can be seen as a vector space over $\mathbb{F}_q$.

If X is an algebraic curve over $\mathbb{F}_q$ and α is an automorphism of X, then we say that α is defined over $\mathbb{F}_q$ provided α commutes with the Frobenius map Frob_q. The automorphisms of X which are defined over $\mathbb{F}_q$ form a subgroup of $\mathrm{Aut}(X)$.

2.5 Algebraic-geometry codes

Algebraic-geometry (AG) codes are linear codes constructed from algebraic curves defined over a finite field $\mathbb{F}_q$. The best known such general construction was originally introduced by Goppa, see [Gop88]. It provides linear codes from certain rational functions whose poles are prescribed by a given $\mathbb{F}_q$-rational divisor G, by evaluating them at some set of $\mathbb{F}_q$-rational places disjoint from $\mathrm{supp}(G)$. The dual to such a code can be obtained by computing residues of differential forms. The former are the *functional* codes, and the latter are the *differential* codes.

Let X be a smooth plane curve defined over the finite field $\mathbb{F}_q$. Write $D = Q_1 + \ldots + Q_n$ for the $\mathbb{F}_q$-rational places $Q_1, \ldots, Q_n$. Let G be another divisor of $\mathbb{F}_q(X)$ whose support $\mathrm{supp}(G)$ contains none of the places Q_i with $1 \le i \le n$. For any function $h \in \mathscr{L}(G)$, the *evaluation* of h at D is given by

$$\mathrm{ev}_D(h) = (h(Q_1), \ldots h(Q_n)).$$

This defines the *evaluation map* $\mathrm{ev}_D : \mathscr{L}(G) \to \mathbb{F}_q^n$ which is $\mathbb{F}_q$-linear and also injective when $n > \deg(G)$. Therefore, its image is a subspace of the vector space $\mathbb{F}_q^n$, or equivalently, an AG $[n,k,d]$-code where $d \ge n - \deg(G)$ and if $\deg(G) > 2g-2$ then $k = \deg(G) + 1 - g$. Such a code is the *functional* code

$$C_L(D,G) = \{(h(Q_1), \ldots, h(Q_n)) \mid h \in \mathscr{L}(G)\}$$

with *designed minimum distance* $n - \deg(G)$. The dual code

$$C_\Omega(D,G) = \{(\mathrm{res}_{Q_1}(dh), \ldots, \mathrm{res}_{Q_1}(dh)) \mid dh \in \Omega(G-D)\}$$

of $C_L(D,G)$ is named a *differential code*. The differential code $C_\Omega(D,G)$ is a $[n, \ell(G-D) - \ell(G) + \deg D, d]$-code with $d \ge \deg(G) - (2g-2)$, and its designed minimum distance is $\deg(G) - (2g-2)$.

Typically the divisor G is taken to be a multiple mP of a single place P of degree one. Such codes are the *one-point* codes, and have been extensively investigated. It has been shown however that AG-codes with better parameters than the comparable one-point Hermitian code may be obtained by allowing the divisor G to be more general, see [MM05] and the references therein.

2.6 Hermitian curves over finite fields

This package implements places, divisors and Riemann-Roch spaces of the *Hermitian curve* H_q defined over $\mathbb{F}_{q^2}$. We quote the most important geometric and combinatorial properties of H_q, the references are [Hir98] and [HP73]. In the projective plane $PG(2,\mathbb{F}_{q^2})$ equipped with homogeneous coordinates $(X:Y:Z)$, a canonical form of H_q is $X^{q+1}-Y^qZ-YZ^q=0$ so that

$$H_q : X^{q+1} = Y^q + Y$$

in the affine equation. Every $\mathbb{F}_{q^2}$-rational place of the function field $\mathbb{F}_{q^2}(H_q)$ of H_q corresponds to a point of H_q in $PG(2,\mathbb{F}_{q^2})$, and this holds true for the degree one places of the constant field extension $\mathbb{F}_{q^{2k}}(H_q)$ which correspond to the points of H_q in $PG(2,\mathbb{F}_{q^{2k}})$. Moreover, a place P of degree $r>1$ of $\mathbb{F}_{q^2}(H_q)$ is represented by a divisor $P_1+P_2+\ldots+P_r$ of the constant field extension $\mathbb{F}_{q^{2r}}(H_q)$ where P_i are degree one places of $\mathbb{F}_{q^{2r}}(H_q)$ with $P_i = P_1^{\mathrm{Frob}^i_{q^2}}$ for $i=0,1,\ldots,r-1$. Furthermore,

$$|H_q(\mathbb{F}_{q^2})| = |H_q(\mathbb{F}_{q^4})| = q^3+1$$

and

$$|H_q(\mathbb{F}_{q^6})| = q^6+1+q^4(q-1),$$

where $H_q(K)$ denotes the set of K-rational points of the projective curve H_q. A line l of $PG(2,\mathbb{F}_{q^2})$ is either a tangent to H_q at an $\mathbb{F}_{q^2}$-rational point of H_q or it meets H_q at $q+1$ distinct $\mathbb{F}_{q^2}$-rational points. In terms of intersection divisors, see \cite[Section 6.2]{HKT_book},

$$I(H_q,l) = (q+1)Q$$

for the point $Q \in H_q(\mathbb{F}_{q^2})$ of tangent l of H_q, and

$$I(H_q,l) = \sum_{i=1}^{q+1} Q_i$$

for the $q+1$ distinct points of intersections $Q_1,\ldots,Q_{q+1}$ of l and H_q.

Through every point $V \in PG(2,\mathbb{F}_{q^2})$ not in $H_q(\mathbb{F}_{q^2})$ there are q^2-q+1 secants and $q+1$ tangents to H_q. The corresponding $q+1$ tangency points are the common points of H_q with the polar line of V relative to the unitary polarity associated to H_q. Let $V=(1:0:0)$. Then the line l_∞ of equation $Z=0$ is tangent at $P_\infty=(0:1:0)$ while another line through V with equation $Y-cZ=0$ is either a tangent or a secant according as c^q+c is 0 or not.

If K is the algebraic closure of $\mathbb{F}_{q^2}$ with $q>2$, then the group of K-automorphisms of the Hermitian curve H_q is the projective unitary group $PGU(3,q)$. In particular, all automorphisms of H_q are defined over $\mathbb{F}_{q^2}$. The automorphism group act doubly transitively on the set of $\mathbb{F}_{q^2}$-rational points.

Chapter 3

How to use the package

3.1 Hermitian curves

The following functions are available:

3.1.1 IsHermitian_Curve

▷ `IsHermitian_Curve(obj)` (Category)

Hermitian curve $H(q)$ is an algebraic curve over an algebraically closed field, having an affine equation $X^{q+1} = Y^q + Y$. The base field of $H(q)$ is $GF(q^2)$.

3.1.2 Hermitian_Curve

▷ `Hermitian_Curve(K, hratfn)` (operation)

returns the corresponding Hermitian curve $H(q)$ over the algebraic closure of the field `K`. The indeterminates X,Y of `hratfn` generate the corresponding Hermitian function field $K(X,Y)$ such that $X^{q+1} = Y^q + Y$. `K` must be a finite field of square order. The points of $H(q)$ are either affine $P(a,b)$ satisfying $a^{q+1} = b^q + b$, or the infinite point `[ infinity ]`. One can use the `in` operation to test if a point lies on the Hermitian curve.

3.1.3 IndeterminatesOfHermitian_Curve

▷ `IndeterminatesOfHermitian_Curve(Hq)` (function)

returns the indeterminates of the function field of the Hermitian curve `C`.

3.1.4 UnderlyingField

▷ `UnderlyingField(Hq)` (attribute)

The underlying field of a Hermitian curve is the field of coefficients of the corresponding algebraic function field, it is a finite field of square order.

3.1.5 RandomPlaceOfGivenDegreeOfHermitian_Curve

▷ `RandomPlaceOfGivenDegreeOfHermitian_Curve(Hq, d)` (operation)

returns a random place of degree *d* of the Hermitian curve *Hq*, that is, a place defined over the field $GF(q^{2d})$. Notice that the place at infinity is has degree 1.

Example

```
gap> Y:=Indeterminate(GF(9),"Y");
Y
gap> C:=Hermitian_Curve(GF(9),Y);
<GZ curve over GF(9) with indeterminate Y>
gap> aut:=AutomorphismGroup(C);
<group of GZ curve automorphisms of size 720>
gap> Random(aut);
Hermitian_CurveAut([ [ Z(3)^0, Z(3^2)^3 ], [ Z(3^2)^5, Z(3) ] ])
```

3.1.6 FrobeniusAutomorphismOfHermitian_Curve

▷ `FrobeniusAutomorphismOfHermitian_Curve(Hq)` (attribute)

returns the Frobenius automorphism of the underlying field of the Hermitian curve *Hq*. More precisely, the output is an AC-Frobenius automorphism in the sense of the package OnAlgClosure, acting on the algebraic closure of the underlying finite field.

3.1.7 IsHermitian_CurveAutomorphism

▷ `IsHermitian_CurveAutomorphism(obj)` (Category)

With automorphisms of an algebraic curve C one means the automorphisms of the corresponding algebraic function field $K(C)$. For Hermitian curves over finite fields, the algebraic function field is the field $K(t)$ of rational functions in one indeterminate. $Aut(K(t))$ consists of fractional linear mappings $t \mapsto \frac{a+bt}{c+dt}$, where $ad-bc \neq 0$. Hence, $Aut(K(t)) \cong PGL(2,K)$.

With fixed Frobenius automorphism $\Phi : x \mapsto x^q$, we can speak of $GF(q)$-rational automorphisms, or, automorphisms defined over $GF(q)$. These form a subgroup isomorphic to $PGL(2,q)$, having a faithful permutation representation of the set $GF(q) \cup \{\infty\}$ of $GF(q)$-rational places.

3.1.8 Hermitian_CurveAutomorphism

▷ `Hermitian_CurveAutomorphism(mat)` (operation)

Returns: the automorphism $t \mapsto \frac{a+bt}{c+dt}$ of the Hermitian curve, where *M* is the nonsingular 2×2 matrix $\begin{pmatrix} a & c \\ b & d \end{pmatrix}$.

3.1.9 AutomorphismGroup

▷ `MatrixGroupToHermitian_CurveAutGroup(matgr, C)` (function)

Returns: the GZ curve automorphism group G corresponding to the matrix group *matgr*.

The permutation action of *matgr* on the set of rational places of *C* is stored as a nice monomorphism of G. ▷ AutomorphismGroup(*C*) (operation)

Returns: the automorphism group of the Hermitian curve *C*. The elements are Hermitian automorphisms. The group is isomorphic to $PGL(2,q)$, where $GF(q)$ is the underlying field of *C*.

3.2 Hermitian divisors

The following functions are available:

3.2.1 IsHermitian_Divisor

▷ IsHermitian_Divisor(*obj*) (Category)

A Hermitian divisor is a divisor of an algebraic function field of the Hermitian curve $H(q) : X^{q+1} = Y^q + Y$. Hermitian divisors form an additive commutative group.

3.2.2 Hermitian_DivisorConstruct

▷ Hermitian_DivisorConstruct(*Hq, pts, ords*) (function)

returns the Hermitian divisor over *Hq* with points from *pts* and corresponding orders from *ords*. It checks the input.

3.2.3 Hermitian_Divisor

▷ Hermitian_Divisor(*Hq, pts, ords*) (operation)
▷ Hermitian_Divisor(*Hq, pairs*) (operation)

returns the corresponding Hermitian divisor over the Hermitian curve *Hq*. The list *pts* must be points of *Hq*; the infinite point is `[ infinity ]`. The list *ords* contains the respective orders. The elements of the list *pairs* are the point-order pairs.

3.2.4 Hermitian_Place

▷ Hermitian_Place(*Hq, pt*) (operation)

returns the corresponding place of the Hermitian curve *Hq*, where *pt* is either an affine point *Hq*, or the infinite point is `[ infinity ]`.

3.2.5 ZeroHermitian_Divisor

▷ ZeroHermitian_Divisor(*Hq*) (operation)

returns the zero divisor over the Hermitian curve *Hq*.

3.2.6 IsRationalHermitian_Divisor

▷ `IsRationalHermitian_Divisor(D)` (attribute)

Returns true if *D* is invariant under the Frobenius automorphism of the underlying Hermitian curve.

3.2.7 UnderlyingField

▷ `UnderlyingField(D)` (attribute)

The underlying field of a Hermitian divisor is the field of coefficients of the corresponding Hermitian curve.

3.2.8 Support

▷ `Support(D)` (attribute)

The support of a Hermitian divisor is the set of points with nonzero orders.

3.2.9 Valuation

▷ `Valuation(D, pt)` (operation)

The valuation of a Hermitian divisor *D* at the point or place *pt* is its corresponding order.

3.2.10 PrincipalHermitian_Divisor

▷ `PrincipalHermitian_Divisor(Hq, f)` (operation)

returns the principal divisor of the rational function *f* of the Hermitian curve *Hq*.

3.2.11 SupremumHermitian_Divisor

▷ `SupremumHermitian_Divisor(D1, D2)` (function)

returns the place-wise maximum of the orders of *D1* and *D2*.

3.2.12 InfimumHermitian_Divisor

▷ `InfimumHermitian_Divisor(D1, D2)` (function)

returns the place-wise minimum of the orders of *D1* and *D2*.

3.2.13 PositivePartOfHermitian_Divisor

▷ `PositivePartOfHermitian_Divisor(D)` (function)

returns the positive part of the divisor *D*.

3.2.14 NegativePartOfHermitian_Divisor

▷ NegativePartOfHermitian_Divisor(*D*) (function)

returns the negative part of the divisor *D*.

Example

```
gap> p1:=Hermitian_Place(C,infinity);
<GZ divisor with support of length 1 over indeterminate Y>
gap> p2:=Hermitian_Place(C,Z(3));
<GZ divisor with support of length 1 over indeterminate Y>
gap> d:=3*p1-4*p2;
<GZ divisor with support of length 2 over indeterminate Y>
gap> Support(d);
[ infinity, Z(3) ]
gap> UnderlyingField(d);
GF(3^2)
gap> Zero(d);
<GZ divisor with support of length 0 over indeterminate Y>
gap> Characteristic(d);
3
gap>
gap> d:=Hermitian_Divisor(C,[Z(27)^2,Z(3),infinity],[5,-1,2]);
<GZ divisor with support of length 3 over indeterminate Y>
gap> Valuation(Z(3),d);
-1
gap> Valuation(Z(3)^2,d);
0
gap>
gap> fr:=AC_FrobeniusAutomorphism(9);
AC_FrobeniusAutomorphism(3^2)
gap> d^fr;
<GZ divisor with support of length 3 over indeterminate Y>
gap> Support(d^fr);
[ infinity, Z(3), Z(3^3)^18 ]
gap> Support(d);
[ infinity, Z(3), Z(3^3)^2 ]
gap>
gap> rf:=Y^8-1;
Y^8-Z(3)^0
gap> List(GF(9),u->Valuation(u,rf));
[ 0, 1, 1, 1, 1, 1, 1, 1, 1 ]
gap> List(GF(9),u->Valuation(u,One(Y)));
[ 0, 0, 0, 0, 0, 0, 0, 0, 0 ]
gap> List(GF(9),u->Valuation(u,Zero(Y)));
[ -infinity, -infinity, -infinity, -infinity, -infinity, -infinity,
  -infinity, -infinity, -infinity ]
gap>
gap>
gap> List(GF(3),u->Valuation(u,One(Y)));
[ 0, 0, 0 ]
gap> List(GF(3),u->Valuation(u,Zero(Y)));
[ -infinity, -infinity, -infinity ]
```

3.3 Hermitian Riemann-Roch spaces

3.3.1 Hermitian_RiemannRochSpaceBasis

▷ `Hermitian_RiemannRochSpaceBasis(D)` (function)

returns a BASIS of the Riemann-Roch space of the Hermitian divisor `D`, which is defined by $\{f \in K[Y] \mid Div(f) \geq -D\}$.

Example

```
gap> a:=RandomPlaceOfHermitian_Curve(C,4);
<GZ divisor with support of length 1 over indeterminate Y>
gap> fr:=FrobeniusAutomorphismOfHermitian_Curve(C);
AC_FrobeniusAutomorphism(3^2)
gap> d:=Sum(AC_FrobeniusAutomorphismOrbit(fr,a));
<GZ divisor with support of length 4 over indeterminate Y>
gap> IsRationalHermitian_Divisor(d);
true
gap>
gap> Hermitian_RiemannRochSpaceBasis(3*d);
[ Z(3)^0/(Y^12+Y^9+Z(3^2)^2*Y^6+Z(3^2)^3*Y^3+Z(3^2)^2),
  Y/(Y^12+Y^9+Z(3^2)^2*Y^6+Z(3^2)^3*Y^3+Z(3^2)^2),
  Y^2/(Y^12+Y^9+Z(3^2)^2*Y^6+Z(3^2)^3*Y^3+Z(3^2)^2),
  Y^3/(Y^12+Y^9+Z(3^2)^2*Y^6+Z(3^2)^3*Y^3+Z(3^2)^2),
  Y^4/(Y^12+Y^9+Z(3^2)^2*Y^6+Z(3^2)^3*Y^3+Z(3^2)^2),
  Y^5/(Y^12+Y^9+Z(3^2)^2*Y^6+Z(3^2)^3*Y^3+Z(3^2)^2),
  Y^6/(Y^12+Y^9+Z(3^2)^2*Y^6+Z(3^2)^3*Y^3+Z(3^2)^2),
  Y^7/(Y^12+Y^9+Z(3^2)^2*Y^6+Z(3^2)^3*Y^3+Z(3^2)^2),
  Y^8/(Y^12+Y^9+Z(3^2)^2*Y^6+Z(3^2)^3*Y^3+Z(3^2)^2),
  Y^9/(Y^12+Y^9+Z(3^2)^2*Y^6+Z(3^2)^3*Y^3+Z(3^2)^2),
  Y^10/(Y^12+Y^9+Z(3^2)^2*Y^6+Z(3^2)^3*Y^3+Z(3^2)^2),
  Y^11/(Y^12+Y^9+Z(3^2)^2*Y^6+Z(3^2)^3*Y^3+Z(3^2)^2),
  Y^12/(Y^12+Y^9+Z(3^2)^2*Y^6+Z(3^2)^3*Y^3+Z(3^2)^2) ]
gap> ForAll(last,x->x=x^fr);
true
```

3.4 Hermitian AG-codes

The following functions are available:

3.4.1 IsHermitian_Code

▷ `IsHermitian_Code(obj)` (Category)
▷ `IsHermitian_FunctionalCode(obj)` (Category)
▷ `IsHermitian_DifferentialCode(obj)` (Category)

A Hermitian code is an algebraic-geometric (AG) code defined on the Hermitian curve of equation $X^{q+1} = Y^q + Y$. AG-codes are either of functional or of differential type.

3.4.2 GeneratorMatrixOfFunctionalHermitian_CodeNC

▷ `GeneratorMatrixOfFunctionalHermitian_CodeNC(G, pls)` (function)

returns the generator matrix of the functional AG code $C_L(D,G)$, where D is the sum of the degree one places in the list `pls`. The support of `G` must be disjoint from `pls`.

3.4.3 Hermitian_FunctionalCode

▷ `Hermitian_FunctionalCode(G, D)` (operation)
▷ `Hermitian_FunctionalCode(G)` (operation)

returns the functional AG code $C_L(D,G) = \{(f(P_1),\dots,f(P_n)) \mid f \in L(G)\}$. D and G are rational divisors of the Hermitian curve C. $D = P_1 + \cdots + D_n$, where $P_1,\dots,P_n$ are degree one places of C. The supports of D and G are disjoint. If D is not given then it is the sum of affine rational places of $H(q)$, not contained in the support of G. By the Riemann-Roch theorem, functional codes have dimension at least $\deg(G)+1-g$, with equality if $\deg(G) > 2g-2$.

3.4.4 Hermitian_DifferentialCode

▷ `Hermitian_DifferentialCode(G, D)` (operation)
▷ `Hermitian_DifferentialCode(G)` (operation)

returns the differential AG code $C_\Omega(D,G) = \{res_{P_1}(\omega),\dots,res_{P_n}(\omega) \mid \omega \in \Omega(G-D)\}$. D and G are rational divisors of the Hermitian curve C. $D = P_1 + \cdots + D_n$, where $P_1,\dots,P_n$ are degree one places of C. The supports of D and G are disjoint. If D is not given then it is the sum of affine rational places of $H(q)$, not contained in the support of G. By the Riemann-Roch theorem, functional codes have dimension $\deg(G)+1-g$. The differential code is the dual of the corresponding functional code. By the Riemann-Roch theorem, differential codes have dimension at least $n-\deg(G)-1+g$, with equality if $\deg(G) > 2g-2$.

3.4.5 Length

▷ `Length(C)` (attribute)

returns the length of the AG code `C`.

3.4.6 GeneratorMatrixOfHermitian_Code

▷ `GeneratorMatrixOfHermitian_Code(C)` (attribute)

returns the generator matrix of the AG code `C` in CVEC matrix format.

3.4.7 DesignedMinimumDistance

▷ `DesignedMinimumDistance(C)` (attribute) returns the designed minimum distance δ of the Hermitian AG code `C`. When $\deg(G) \geq 2g-2$, then the general formulas for δ are as follows. For the functional code $C_L(D,G)$, $\delta = n - \deg(G)$, and for the differential code $C_\Omega(D,G)$, $\delta = \deg(G) - (2g-2)$.

Example

```
gap> code:=Hermitian_FunctionalCode(d);
<[9,5] Hermitian AG-code over GF(3^2)>
gap> Print(code);
Hermitian_FunctionalCode(Hermitian_Divisor(Hermitian_Curve(GF(9),Y),
[ Z(3^8)^302, Z(3^8)^2718, Z(3^8)^3678, Z(3^8)^4782 ],
[ 1, 1, 1, 1 ]),Hermitian_Divisor(Hermitian_Curve(GF(9),Y),
[ 0*Z(3), Z(3)^0, Z(3), Z(3^2), Z(3^2)^2, Z(3^2)^3, Z(3^2)^5,
  Z(3^2)^6, Z(3^2)^7 ],[ 1, 1, 1, 1, 1, 1, 1, 1, 1 ]))
gap> DesignedMinimumDistance(code);
5
```

3.4.8 Hermitian_DecodeToCodeword

▷ `Hermitian_DecodeToCodeword(C, w)` (operation)

Let δ be the designed minimum distance of `C`, and define $t = [(\delta - 1 - g)/2]$. If there is a codeword $c \in C$ with $d(c,w) \leq t$ then c is returned. Otherwise, the output is `fail`.

The decoding algorithm is from [Hoholdt-Pellikaan 1995]. The function `Hermitian_DECODER_DATA` precomputes two matrices which are stored as attributes of the AG code. The decoding consists of solving linear equations.

Example

```
gap> q:=5^3;
125
gap> # construct the curve and the divisors
gap> Y:=Indeterminate(GF(q),"Y");
Y
gap> C:=Hermitian_Curve(GF(q),Y);
<GZ curve over GF(125) with indeterminate Y>
gap> P_infty:=Hermitian_1PointDivisor(C,infinity);
<GZ divisor with support of length 1 over indeterminate Y>
gap>
gap> fr:=FrobeniusAutomorphismOfHermitian_Curve(C);
AC_FrobeniusAutomorphism(5^3)
gap> P4:=Sum(AC_FrobeniusAutomorphismOrbit(fr,RandomPlaceOfHermitian_Curve(C,4)));
<GZ divisor with support of length 4 over indeterminate Y>
gap> G:=5*P4+7*P_infty;
<GZ divisor with support of length 5 over indeterminate Y>
gap> Degree(G);
27
gap>
gap> len:=90;
90
gap> D:=Sum([1..len],i->Hermitian_1PointDivisor(C,Elements(GF(q))[i]));
<GZ divisor with support of length 90 over indeterminate Y>
gap>
gap> # construct the AG differential code
gap> agcode:=Hermitian_DifferentialCode(G,D);
```

```
<[90,62] Hermitian AG-code over GF(5^3)>
gap> DesignedMinimumDistance(agcode);
29
gap> Length(agcode)-Degree(G)-1;
62
gap>
gap> # test codeword generation
gap> t:=Int((DesignedMinimumDistance(agcode)-1)/2);
14
gap> sent:=Random(agcode);;
gap> err:=RandomVectorOfGivenWeight(GF(q),Length(agcode),t);;
gap> received:=sent+err;;
gap>
gap> # decoding
gap> sent_decoded:=Hermitian_DecodeToCodeword(agcode,received);
<cvec over GF(5,3) of length 90>
gap> sent=sent_decoded;
true
```

3.5 Utilities for Hermitian AG-codes

3.5.1 RestrictVectorSpace

▷ `RestrictVectorSpace(V, F)` (function)

Let K be a field and V a linear subspace of K^n. The restriction of V to the field F is the intersection $V \cap F^n$.

3.5.2 UPolCoeffsToSmallFieldNC

▷ `UPolCoeffsToSmallFieldNC(f, q)` (function)

This non-checking function returns the same polynomial as `f`, making sure that the coefficients are in $GF(q)$.

3.5.3 RandomVectorOfGivenWeight

▷ `RandomVectorOfGivenWeight(F, n, k)` (function)

returns a random vector of F^n of Hamming weight k. ▷ `RandomVectorOfGivenDensity(F, n, delta)` (function)

returns a random vector of F^n in which the density of nonzero elements is approximately δ. ▷ `RandomBinaryVectorOfGivenWeight(n, k)` (function)

returns a random vector of $GF(2)^n$ of Hamming weight k. ▷ `RandomBinaryVectorOfGivenDensity(n, delta)` (function) returns a random vector of $GF(2)^n$ in which the density of nonzero elements is approximatively δ.

Chapter 4

An example: BCH codes as Hermitian AG-codes

The following example constructs BCH codes as Hermitian AG-codes.

Example

```
gap> my_BCH:=function(n,l,delta,F)
>         local q,m,r,s,beta,Y,C,D_beta,P_0,P_infty,agcode;
>         #
>         q:=Size(F);
>         m:=OrderMod(q,n);
>         beta:=Z(q^m)^((q^m-1)/n);
>         #
>         Y:=Indeterminate(F,"Y");
>         C:=Hermitian_Curve(GF(q^m),Y);
>         D_beta:=Sum([0..n-1],i->Hermitian_1PointDivisor(C,beta^i));
>         P_0:=Hermitian_1PointDivisor(C,0);
>         P_infty:=Hermitian_1PointDivisor(C,infinity);
>         #
>         r:=l-1;
>         s:=n+1-delta-l;
>         agcode:=Hermitian_FunctionalCode(r*P_0+s*P_infty,D_beta);
>         #
>         return RestrictVectorSpace(agcode,F);
> end;
function( n, l, delta, F ) ... end
gap>
gap> ####
gap>
gap> q:=2;
2
gap> n:=35;
35
gap> l:=1;
1
gap> delta:=5;
5
gap>
gap>
gap> C0:=BCHCode(n,l,delta,GF(q)); time;
```

```
a cyclic [35,11,5]8..13 BCH code, delta=5, b=1 over GF(2)
24
gap> C1:=my_BCH(n,l,delta,GF(q)); time;
<vector space over GF(2), with 11 generators>
364
gap>
gap> Collected(List(C0,x->Number(x,y->IsOne(y))));
[ [ 0, 1 ], [ 5, 7 ], [ 7, 5 ], [ 10, 56 ], [ 13, 105 ], [ 14, 10 ],
  [ 15, 105 ], [ 16, 385 ], [ 17, 350 ], [ 18, 350 ], [ 19, 385 ],
  [ 20, 105 ], [ 21, 10 ], [ 22, 105 ], [ 25, 56 ], [ 28, 5 ],
  [ 30, 7 ], [ 35, 1 ] ]
gap> Collected(List(C1,x->Number(x,y->IsOne(y))));
[ [ 0, 1 ], [ 5, 7 ], [ 7, 5 ], [ 10, 56 ], [ 13, 105 ], [ 14, 10 ],
  [ 15, 105 ], [ 16, 385 ], [ 17, 350 ], [ 18, 350 ], [ 19, 385 ],
  [ 20, 105 ], [ 21, 10 ], [ 22, 105 ], [ 25, 56 ], [ 28, 5 ],
  [ 30, 7 ], [ 35, 1 ] ]
gap>
gap> SetDesignedMinimumDistance(C1,delta);
gap> DesignedMinimumDistance(C1);
5
```

References

[Gop88] V. D. Goppa. *Geometry and codes*, volume 24 of *Mathematics and its Applications (Soviet Series)*. Kluwer Academic Publishers Group, Dordrecht, 1988. Translated from the Russian by N. G. Shartse. 8

[Hir98] J. W. P. Hirschfeld. *Projective geometries over finite fields*. Oxford Mathematical Monographs. The Clarendon Press, Oxford University Press, New York, second edition, 1998. 9

[HKT08] J. W. P. Hirschfeld, G. Korchmáros, and F. Torres. *Algebraic curves over a finite field*. Princeton Series in Applied Mathematics. Princeton University Press, Princeton, NJ, 2008. 6

[HP73] Daniel R. Hughes and Fred C. Piper. *Projective planes*. Springer-Verlag, New York-Berlin, 1973. Graduate Texts in Mathematics, Vol. 6. 9

[HP95] Tom Høholdt and Ruud Pellikaan. On the decoding of algebraic-geometric codes. *IEEE Trans. Inform. Theory*, 41(6, part 1):1589–1614, 1995. Special issue on algebraic geometry codes. 6

[MM05] Gretchen L. Matthews and Todd W. Michel. One-point codes using places of higher degree. *IEEE Trans. Inform. Theory*, 51(4):1590–1593, 2005. 8

[Sti09] Henning Stichtenoth. *Algebraic function fields and codes*, volume 254 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, second edition, 2009. 6

Index