

# 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: [nagy@math.u-szeged.hu](mailto:nagy@math.u-szeged.hu)  
Homepage: <http://www.math.u-szeged.hu/~nagy/>

**Sabira El Khalfaoui** Email: [sabira@math.u-szeged.hu](mailto:sabira@math.u-szeged.hu)

## **Copyright**

© 2019 by Gábor P. Nagy

HERmitian package is free software; you can redistribute it and/or modify it under the terms of the [GNU General Public License](#) as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

## **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

<b>1</b>	<b>Introduction</b>	<b>4</b>
1.1	Unpacking the HERmitian Package . . . . .	4
1.2	Loading the HERmitian Package . . . . .	5
1.3	Testing the HERmitian Package . . . . .	5
<b>2</b>	<b>Mathematical background</b>	<b>6</b>
2.1	Algebraic curves, places, divisors . . . . .	6
2.2	Function fields and Riemann-Roch spaces . . . . .	6
2.3	Automorphisms of algebraic curves . . . . .	7
2.4	Algebraic plane curves over finite fields . . . . .	7
2.5	Algebraic-geometry codes . . . . .	8
2.6	Hermitian curves over finite fields . . . . .	8
<b>3</b>	<b>How to use the package</b>	<b>10</b>
3.1	Hermitian curves . . . . .	10
3.2	Automorphisms of Hermitian curves . . . . .	12
3.3	Hermitian divisors . . . . .	12
3.4	Hermitian Riemann-Roch spaces . . . . .	15
3.5	Hermitian AG-codes . . . . .	16
3.6	Utilities for Hermitian AG-codes . . . . .	19
<b>4</b>	<b>An example: BCH type codes as Hermitian AG codes (???)</b>	<b>21</b>
	<b>References</b>	<b>22</b>
	<b>Index</b>	<b>23</b>

# 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:

Example `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, download the package archive from the [package webpage](#), 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 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  $\mathcal{X}$  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  $D = n_1 P_1 + \dots + n_k P_k$  with integers  $n_1, \dots, n_k$  and places  $P_1, \dots, P_k$ . The degree of  $D$  is  $n_1 + \dots + n_k$ . 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  $h \in K(X)$ ,  $\text{Div}(h)$  stands for the principal divisor associated with  $h$  while  $\text{Div}(h)_0$  and  $\text{Div}(h)_\infty$  for its zero and pole divisor. Furthermore, for every separable function  $h \in K(X)$ ,  $dh$  is the exact differential arising from  $h$ , and  $\Omega$  denotes the set of all these differentials. Also,  $\text{res}_P(dh)$  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

$$\mathcal{L}(A) = \{h \in K(X) \setminus \{0\} \mid \text{Div}(h) \succeq -A\} \cup \{0\}.$$

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

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

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  $\alpha$  of  $X$ , there are polynomials  $u, v, w \in K[X, Y]$  such that

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

Substituting formal power series in  $\alpha$ , we obtain an action of  $\alpha$  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*  $\text{Frob}_q : x \mapsto x^q$  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  $\text{Frob}_q$ -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

$$P + P^{\text{Frob}_q} + P^{\text{Frob}_q^2} + \dots + P^{\text{Frob}_q^{r-1}}$$

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  $\mathcal{L}(A)$  has a basis which consists of  $\mathbb{F}_q$ -rational elements of the function field of  $X$ . Hence, we can view  $\mathcal{L}(A)$  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  $\alpha$  is an automorphism of  $X$ , then we say that  $\alpha$  is defined over  $\mathbb{F}_q$  provided  $\alpha$  commutes with the Frobenius map  $\text{Frob}_q$ . The automorphisms of  $X$  which are defined over  $\mathbb{F}_q$  form a subgroup of  $\text{Aut}(X)$ .

## 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  $\text{supp}(G)$ . 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  $D = Q_1 + \dots + Q_n$  for the  $\mathbb{F}_q$ -rational places  $Q_1, \dots, Q_n$ . Let  $G$  be another divisor of  $\mathbb{F}_q(X)$  whose support  $\text{supp}(G)$  contains none of the places  $Q_i$  with  $1 \leq i \leq n$ . For any function  $h \in \mathcal{L}(G)$ , the *evaluation* of  $h$  at  $D$  is given by

$$\text{ev}_D(h) = (h(Q_1), \dots, h(Q_n)).$$

This defines the *evaluation map*  $\text{ev}_D : \mathcal{L}(G) \rightarrow \mathbb{F}_q^n$  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  $d \geq n - \deg(G)$  and if  $\deg(G) > 2g - 2$  then  $k = \deg(G) + 1 - g$ . Such a code is the *functional* code

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

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

$$C_\Omega(D, G) = \{(\text{res}_{Q_1}(dh), \dots, \text{res}_{Q_n}(dh)) \mid dh \in \Omega(G - D)\}$$

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  $d \geq \deg(G) - (2g - 2)$ , 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 refer-



ences 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  $X^{q+1} - Y^q Z - YZ^q = 0$  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  $P_1 + P_2 + \dots + P_r$  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  $P_i = P_1^{\text{Frob}_{q^2}^i}$  for  $i = 0, 1, \dots, r-1$ . 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  $Q_1, \dots, Q_{q+1}$  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  $l_\infty$  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  $\mathcal{C}$ .

#### 3.1.4 Genus

▷ `Genus(Hq)` (attribute)

The genus of the Hermitian curve  $H(q)$  is  $q(q-1)/2$ .

#### 3.1.5 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.

### 3.1.6 AllRationalAffinePlacesOfHermitian\_Curve

▷ AllRationalAffinePlacesOfHermitian\_Curve( $Hq$ ) (operation)

**Returns:** All rational affine places of the Hermitian curve  $Hq$ . The number of rational affine places of  $H(q)$  is  $q^3$ .

### 3.1.7 AllRationalPlacesOfHermitian\_Curve

▷ AllRationalPlacesOfHermitian\_Curve( $Hq$ ) (operation)

**Returns:** All rational places of the Hermitian curve  $Hq$ . Here, a place is a 1-point divisor of degree one, that is, defined over  $GF(q^2)$ . Notice that the place at infinity is rational. The number of rational places of  $H(q)$  is  $q^3 + 1$ .

### 3.1.8 RandomRationalPlaceOfHermitian\_Curve

▷ RandomRationalPlaceOfHermitian\_Curve( $Hq$ ) (operation)

**Returns:** A random rational place of the Hermitian curve  $Hq$ . Here, a place is a 1-point divisor of degree one, that is, defined over  $GF(q^2)$ . Notice that the place at infinity is rational.

### 3.1.9 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 has degree 1.

Example

```
gap> q:=5;
5
gap> Y:=HermitianIndeterminates(GF(q^2),"Y1","Y2");
[ Y1, Y2 ]
gap> Hq:=Hermitian_Curve(Y[1]);
<Hermitian curve over GF(25) with indeterminates [ Y1, Y2 ]>
gap>
gap> UnderlyingField(Hq);
GF(5^2)
gap> p:=RandomPlaceOfGivenDegreeOfHermitian_Curve(Hq,3);
<Hermitian place [ Z(5^6)^12002, Z(5^6)^14911 ] over indeterminates [ Y1, Y2 ]>
gap> p in Hq;
true
gap> [ infinity ] in Hq;
true
gap> [0,0] in Hq;
false
gap> Z(q)*[0,0] in Hq;
true
gap> Size( AllRationalPlacesOfHermitian_Curve(Hq) );
126
```

## 3.2 Automorphisms of Hermitian curves

### 3.2.1 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.

### 3.2.2 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 a Hermitian curve  $H(q)$  over a finite field,  $Aut(GF(q)(H(q)))$  is isomorphic to the projective general linear group  $PGU(3, q)$ . In particular, an automorphism of  $H(q)$  can be represented by a  $3 \times 3$  unitary matrix over  $GF(q^2)$ .

### 3.2.3 Hermitian\_CurveAutomorphism

▷ `Hermitian_CurveAutomorphism(Hq, mat)` (operation)

**Returns:** The automorphism of the Hermitian curve  $H(q)$ , given by the unitary matrix  $mat$ .

### 3.2.4 AutomorphismGroup

▷ `UnitaryGroupToHermitian_CurveAutGroup(matgr, Hq)` (function)

**Returns:** the group  $G$  of automorphisms of the Hermitian curve  $Hq$ , which correspond to the unitary group  $matgr$ .

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

**Returns:** The automorphism group of the Hermitian curve  $Hq$ . The elements are Hermitian curve automorphisms. The group is isomorphic to  $PPG(3, q)$ , where  $GF(q^2)$  is the underlying field of  $Hq$ .

Example

```
gap> aut:=AutomorphismGroup(Hq);
<group of Hermitian curve automorphisms of size 378000>
gap> Random(aut);
Hermitian_CurveAut([ [ Z(5)^0, Z(5^2)^11, Z(5^2)^19 ], [ Z(5^2)^20, Z(5^2)^16, Z(5^2)^21 ], [ Z(5^2)^21, Z(5^2)^16, Z(5^2)^20 ] ])
```

## 3.3 Hermitian divisors

The following functions are available:

### 3.3.1 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.

### 3.3.2 IsHermitian\_Place

▷ `IsHermitian_Place(obj)` (filter)

In this implementation, a Hermitian place is a Hermitian divisor of degree one and support length one.

### 3.3.3 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.

### 3.3.4 IndeterminatesOfHermitian\_Divisor

▷ `IndeterminatesOfHermitian_Divisor(D)` (function)

**Returns:** The pair of indeterminates of the function field of the Hermitian divisor  $D$ .

### 3.3.5 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.

### 3.3.6 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 ]`.

### 3.3.7 ZeroHermitian\_Divisor

▷ `ZeroHermitian_Divisor(Hq)` (operation)

**Returns:** The zero divisor over the Hermitian curve  $Hq$ .

### 3.3.8 SupremumHermitian\_Divisor

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

**Returns:** The place-wise maximum of the orders of  $D1$  and  $D2$ .

### 3.3.9 InfimumHermitian\_Divisor

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

**Returns:** The place-wise minimum of the orders of  $D1$  and  $D2$ .

### 3.3.10 PositivePartOfHermitian\_Divisor

- ▷ `PositivePartOfHermitian_Divisor( $D$ )` (function)  
**Returns:** The positive part of the divisor  $D$ .

### 3.3.11 NegativePartOfHermitian\_Divisor

- ▷ `NegativePartOfHermitian_Divisor( $D$ )` (function)  
**Returns:** The negative part of the divisor  $D$ .

### 3.3.12 UnderlyingField

- ▷ `UnderlyingField( $D$ )` (attribute)

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

### 3.3.13 Support

- ▷ `Support( $D$ )` (attribute)

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

### 3.3.14 Valuation

- ▷ `Valuation( $D$ ,  $pt$ )` (operation)

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

### 3.3.15 Value

- ▷ `Value( $f$ ,  $pl$ )` (operation)  
**Returns:** The value of a Hermitian rational function  $f$  at the place  $pl$ .

### 3.3.16 IsRationalHermitian\_Divisor

- ▷ `IsRationalHermitian_Divisor( $D$ )` (attribute)  
**Returns:** True if  $D$  is invariant under the Frobenius automorphism of the underlying Hermitian curve.

### 3.3.17 PrincipalHermitian\_Divisor

- ▷ `PrincipalHermitian_Divisor( $Hq$ ,  $f$ )` (operation)  
**Returns:** The principal divisor of the rational function  $f$  of the Hermitian curve  $Hq$ .

### 3.3.18 IsInfiniteHermitian\_Place

▷ IsInfiniteHermitian\_Place( $p$ )

(attribute)

**Returns:** True if  $p$  is a Hermitian place at the infinite line.

Example

```
gap> p_infty:=Hermitian_Place(Hq,[infinity]);
<Hermitian place [ infinity ] over indeterminates [ Y1, Y2 ]>
gap> d:=3*p_infty-4*p;
<Hermitian divisor with support of length 2 over indeterminates [ Y1, Y2 ]>
gap> Support(d);
[ [ infinity ], [ Z(5^6)^12002, Z(5^6)^14911 ] ]
gap> UnderlyingField(d);
GF(5^2)
gap> Zero(d);
<Hermitian divisor with support of length 0 over indeterminates [ Y1, Y2 ]>
gap> Characteristic(d);
5
gap>
gap> Valuation(d,p);
-4
gap> Valuation(d,[1,2]);
0
gap>
gap> fr:=FrobeniusAutomorphismOfHermitian_Curve(Hq);
AC_FrobeniusAutomorphism(5^2)
gap> d^fr;
<Hermitian divisor with support of length 2 over indeterminates [ Y1, Y2 ]>
gap> Support(d^fr);
[ [ infinity ], [ Z(5^6)^3194, Z(5^6)^13423 ] ]
gap> Support(d);
[ [ infinity ], [ Z(5^6)^12002, Z(5^6)^14911 ] ]
```

## 3.4 Hermitian Riemann-Roch spaces

### 3.4.1 Hermitian\_RiemannRochSpaceBasis

▷ Hermitian\_RiemannRochSpaceBasis( $D$ )

(function)

**Returns:** A BASIS of the Riemann-Roch space of the Hermitian divisor  $D$ , which is defined by  $\{f \in K[Y] \mid \text{Div}(f) \geq -D\}$ .

Example

```
gap> a:=RandomPlaceOfGivenDegreeOfHermitian_Curve(Hq,3);
<Hermitian place [ Z(5^6)^5885, Z(5^6)^13071 ] over indeterminates [ Y1, Y2 ]>
gap> fr:=FrobeniusAutomorphismOfHermitian_Curve(Hq);
AC_FrobeniusAutomorphism(5^2)
gap> d:=Sum(AC_FrobeniusAutomorphismOrbit(fr,a));
<Hermitian divisor with support of length 3 over indeterminates [ Y1, Y2 ]>
gap> IsRationalHermitian_Divisor(d);
true
gap>
gap> bb:=Hermitian_RiemannRochSpaceBasis(5*d);
[ (Y1^3+Z(5^2)*Y1^2+Z(5^2)^17)/(Z(5^2)*Y1^3+Z(5^2)^17*Y1*Y2^2-Y2^3+Z(5^2)^14*Y1*Y2+Z(5^2)^11*Y2^2
```

```

15),
  (Z(5^2)^16*Y1^2+Z(5^2)^23*Y1+Y2+Z(5^2)^5)/(Z(5^2)*Y1^3+Z(5^2)^17*Y1*Y2^2-Y2^3+Z(5^2)^14*Y1*Y2+Z(5^2)^7*Y2+Z(5^2)^15),
  (Z(5^2)^17*Y1^2+Y1*Y2+Z(5^2)^5*Y1+Z(5^2)^21)/(Z(5^2)*Y1^3+Z(5^2)^17*Y1*Y2^2-Y2^3+Z(5^2)^14*Y1*Y2+Z(5^2)^7*Y2+Z(5^2)^15),
  (Y1^2*Y2+Z(5^2)^3*Y1^2+Z(5^2)^21*Y1+Z(5^2)^22)/(Z(5^2)*Y1^3+Z(5^2)^17*Y1*Y2^2-Y2^3+Z(5^2)^14*Y1*Y2+Z(5^2)^7*Y2+Z(5^2)^15),
  (Z(5^2)^23*Y1^2+Y2^2+Z(5^2)^8*Y1+Z(5^2)^13)/(Z(5^2)*Y1^3+Z(5^2)^17*Y1*Y2^2-Y2^3+Z(5^2)^14*Y1*Y2+Z(5^2)^7*Y2+Z(5^2)^15),
  (Y1*Y2^2+Z(5^2)^7*Y1^2+Z(5^2)^13*Y1+Z(5^2)^4)/(Z(5^2)*Y1^3+Z(5^2)^17*Y1*Y2^2-Y2^3+Z(5^2)^14*Y1*Y2+Z(5^2)^7*Y2+Z(5^2)^15),
  (Y2^3+Z(5^2)^5*Y1^2+Z(5^2)^11*Y1+Z(5^2)^14)/(Z(5^2)*Y1^3+Z(5^2)^17*Y1*Y2^2-Y2^3+Z(5^2)^14*Y1*Y2+Z(5^2)^7*Y2+Z(5^2)^15) ]
gap> Size(bb);
7
gap> ForAll(bb,x->x=x^fr);
true
gap> ForAll(bb,x->PrincipalHermitian_Divisor(Hq,x)>=-5*d);
true

```

### 3.5 Hermitian AG-codes

The following functions are available:

#### 3.5.1 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.

#### 3.5.2 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  $pls$ .

#### 3.5.3 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  $H(q)$ .  $D = P_1 + \dots + P_n$ , where  $P_1, \dots, P_n$  are degree one places of  $H(q)$ . 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  $\deg(G) > 2g - 2$ .



### 3.5.4 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  $H(q)$ .  $D = P_1 + \dots + P_n$ , where  $P_1, \dots, P_n$  are degree one places of  $H(q)$ . 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  $\deg(G) > 2g - 2$ .

### 3.5.5 UnderlyingCurveOfHermitian\_Code

- ▷ `UnderlyingCurveOfHermitian_Code( $C$ )` (function)  
**Returns:** The underlying Hermitian curve of the AG code  $C$ .

### 3.5.6 UnderlyingField

- ▷ `UnderlyingField( $C$ )` (attribute)

The underlying field of an AG code is its left acting domain.

### 3.5.7 Length

- ▷ `Length( $C$ )` (attribute)  
**Returns:** The length of the AG code  $C$ .

### 3.5.8 GeneratorMatrixOfHermitian\_Code

- ▷ `GeneratorMatrixOfHermitian_Code( $C$ )` (attribute)  
**Returns:** The generator matrix of the AG code  $C$  in CVEC matrix format.

### 3.5.9 DesignedMinimumDistance

- ▷ `DesignedMinimumDistance( $C$ )` (attribute)  
**Returns:** The designed minimum distance  $\delta$  of the Hermitian AG code  $C$ . When  $\deg(G) \geq 2g - 2$ , then the general formulas for  $\delta$  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> Hermitian_FunctionalCode(5*d);
<[125,7] Hermitian AG-code over GF(5^2)>
gap> a:=Sum(AllRationalAffinePlacesOfHermitian_Curve(Hq));
<Hermitian divisor with support of length 125 over indeterminates [ Y1, Y2 ]>
gap> code:=Hermitian_FunctionalCode(8*d,a);
<[125,15] Hermitian AG-code over GF(5^2)>
gap> Print(code);
Hermitian_FunctionalCode(Hermitian_Divisor(Hermitian_Curve(Y1),[ [ Z(5^6)^5885, Z(5^6)^13071 ], [
  [ Z(5^6)^6509, Z(5^6)^14295 ] ],[ 8, 8, 8 ]),Hermitian_Divisor(Hermitian_Curve(Y1),[ [ 0*Z(5),
```

### 3.5.10 Hermitian DecodeToCodeword

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.

```
gap> q:=4;
4
gap> # construct the curve and the divisors
gap> Y:=HermitianIndeterminates(GF(q^2),"Y1","Y2");
[ Y1, Y2 ]
gap> Hq:=Hermitian_Curve(Y[1]);
```

```

<Hermitian curve over GF(16) with indeterminates [ Y1, Y2 ]>
gap> P_infty:=Hermitian_Place(Hq,[infinity]);
<Hermitian place [ infinity ] over indeterminates [ Y1, Y2 ]>
gap>
gap> fr:=FrobeniusAutomorphismOfHermitian_Curve(Hq);
AC_FrobeniusAutomorphism(2^4)
gap> P4:=RandomPlaceOfGivenDegreeOfHermitian_Curve(Hq,5);
<Hermitian place [ Z(2,20)+Z(2,20)^3+Z(2,20)^4+Z(2,20)^6+Z(2,20)^11+Z(2,20)^13+Z(2,20)^15+Z(2,20)^17+
  Z(2,20)+Z(2,20)^4+Z(2,20)^5+Z(2,20)^7+Z(2,20)^8+Z(2,20)^12+Z(2,20)^13+Z(2,20)^15+Z(2,20)^19 ] c
gap> P4:=Sum(AC_FrobeniusAutomorphismOrbit(fr,P4));
<Hermitian divisor with support of length 5 over indeterminates [ Y1, Y2 ]>
gap> G:=5*P4+7*P_infty;
<Hermitian divisor with support of length 6 over indeterminates [ Y1, Y2 ]>
gap> Degree(G);
32
gap>
gap> len:=50;
50
gap> affpts:=AllRationalAffinePlacesOfHermitian_Curve(Hq);
gap> D:=Sum(affpts{[1..len]});
<Hermitian divisor with support of length 50 over indeterminates [ Y1, Y2 ]>
gap>
gap> # construct the AG differential code
gap> Hermitian_DifferentialCode(G);
<[64,37] Hermitian AG-code over GF(2^4)>
gap> agcode:=Hermitian_DifferentialCode(G,D);
<[50,23] Hermitian AG-code over GF(2^4)>
gap> DesignedMinimumDistance(agcode);
22
gap> Length(agcode)-Degree(G)-1;
17
gap>
gap> # test codeword generation
gap> t:=Int((DesignedMinimumDistance(agcode)-1-Genus(G!.curve))/2);
7
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(2,4) of length 50>
gap> sent=sent_decoded;
true

```

## 3.6 Utilities for Hermitian AG-codes

### 3.6.1 InfoHermitian

▷ InfoHermitian

(info class)

An infoclass for the package. Its default value is 0. With `SetInfoLevel(InfoHermitian, 2)` one can get some additional messages about the ongoing computation of Hermitian curves and their codes.

### 3.6.2 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  $V \cap F^n$ .

### 3.6.3 UPolCoeffsToSmallFieldNC

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

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

### 3.6.4 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 approximatively  $\delta$ .

▷ `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  $\delta$ .

## Chapter 4

# An example: BCH type codes as Hermitian AG codes (???)

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

Example

```
gap>
```

# 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

AllRationalAffinePlacesOfHermitian\_Curve, [11](#)  
AllRationalPlacesOfHermitian\_Curve, [11](#)  
AutomorphismGroup, [12](#)  
DesignedMinimumDistance, [17](#)  
FrobeniusAutomorphismOfHermitian\_Curve, [12](#)  
GeneratorMatrixOfFunctionalHermitian\_CodeNC, [16](#)  
GeneratorMatrixOfHermitian\_Code, [17](#)  
Genus, [10](#)  
HERmitian package, [4](#)  
Hermitian\_Curve, [10](#)  
Hermitian\_CurveAutomorphism, [12](#)  
Hermitian\_DecomposeToCodeword, [18](#)  
Hermitian\_DifferentialCode, [17](#)  
Hermitian\_Divisor, [13](#)  
Hermitian\_DivisorConstruct, [13](#)  
Hermitian\_FunctionalCode, [16](#)  
Hermitian\_Place, [13](#)  
Hermitian\_RiemannRochSpaceBasis, [15](#)  
IndeterminatesOfHermitian\_Curve, [10](#)  
IndeterminatesOfHermitian\_Divisor, [13](#)  
InfimumHermitian\_Divisor, [13](#)  
InfoHermitian, [19](#)  
IsHermitian\_Code, [16](#)  
IsHermitian\_Curve, [10](#)  
IsHermitian\_CurveAutomorphism, [12](#)  
IsHermitian\_DifferentialCode, [16](#)  
IsHermitian\_Divisor, [12](#)  
IsHermitian\_FunctionalCode, [16](#)  
IsHermitian\_Place, [13](#)  
IsInfiniteHermitian\_Place, [15](#)  
IsRationalHermitian\_Divisor, [14](#)  
Length, [17](#)  
License, [2](#)  
NegativePartOfHermitian\_Divisor, [14](#)  
PositivePartOfHermitian\_Divisor, [14](#)  
PrincipalHermitian\_Divisor, [14](#)  
RandomBinaryVectorOfGivenDensity, [20](#)  
RandomBinaryVectorOfGivenWeight, [20](#)  
RandomPlaceOfGivenDegreeOfHermitian\_Curve, [11](#)  
RandomRationalPlaceOfHermitian\_Curve, [11](#)  
RandomVectorOfGivenDensity, [20](#)  
RandomVectorOfGivenWeight, [20](#)  
RestrictVectorSpace, [20](#)  
Support, [14](#)  
SupremumHermitian\_Divisor, [13](#)  
UnderlyingCurveOfHermitian\_Code, [17](#)  
UnderlyingField, [10](#), [14](#), [17](#)  
UnitaryGroupToHermitian\_CurveAutGroup, [12](#)  
UPolCoeffsToSmallFieldNC, [20](#)  
Valuation, [14](#)  
Value, [14](#)  
ZeroHermitian\_Divisor, [13](#)