Forms

Sesquilinear and Quadratic

1.2.8

9 July 2022

John Bamberg

Jan De Beule

John Bamberg
Email: bamberg@maths.uwa.edu.au
Homepage: http://school.maths.uwa.edu.au/~bamberg/
Address: School of Mathematics and Statistics
The University of Western Australia
35 Stirling Highway
CrawleyY WA 6009, Perth
Australia

Jan De Beule
Email: jan@debeule.eu
Homepage: http://www.debeule.eu
Address: Department of Mathematics and Data Science
Vrije Universiteit Brussel
Pleinlaan 2
B-1050 Brussel
Belgium

Copyright

Contents

Chapter 1

Introduction

1.1 Philosophy

Forms is a package for computing with sesquilinear and quadratic forms on finite vector spaces. It provides users with the basic algebraic tools to work with classical groups and polar geometries, and enables one to specify a form and its corresponding geometry. The functionality of the package includes:

- the construction of sesquilinear and quadratic forms;
- operations which allow a user to change coordinates, that is, to “change form” and work in an isometric (or similar) formed vector space; and
- a way to determine the form(s) left invariant by a matrix group (up to a scalar).

1.2 Overview over this manual

The next chapter (2) gives some basic examples of the use of this package. In "Background Theory of Forms" (Chapter 3) we revise the basic notions of the theory of sesquilinear and quadratic forms, where we also set the notation and conventions adopted by this package. In "Constructing forms and basic functionality" (Chapter 4), we describe all operations to construct sesquilinear and quadratic forms and basic attributes and properties that do not require morphisms. In "Morphims of forms" (Chapter 5) we revise the basic notions of morphisms of forms, and the classification of sesquilinear and quadratic forms on vector spaces over finite fields. Operations, attributes and properties that are related to the computation of morphisms of forms, are also described in this chapter.

1.3 How to read this manual

We have tried to make this manual pleasant to read for the general reader. So it is inevitable that we will use Greek symbols and simple mathematical formulas. To make these visible in the HTML version of this documentation, you may have to change the default character set of your browser to UTF-8.

1.4 Web resources

- Find Forms on the Packages section of the GAP-website: https://www.gap-system.org/Packages/forms.html.
- Find Forms on its homepage: https://gap-packages.github.io/forms.
- Report bugs, questions and issues on the Forms issue tracker: https://github.com/gap-packages/forms/issues

1.5 Release notes

Version 1.2.1 of Forms contains some changed and extra functionality with relation to trivial forms. The changed and new functionality is described completely in Section 4.9. We gratefully acknowledge the useful feedback of Alice Niemeyer.

In version 1.2.2 of Forms a minor bug, pointed out by John Bamberg, in the code of `IsTotallyIsotropicSubspace` is repaired. On the occasion of the release of the first beta versions of GAP4r5, we changed the names of some global functions such that a name clash becomes unlikely. Version 1.2.2 of Forms is compatible with GAP4r4 and GAP4r5.

Version 1.2.3 contains a new operation `TypeOfForm`. Together with this addition, some parts of the documentation, especially concerning degenerate and singular forms, have been edited. A bug found in the methods for `\^` applicable on a pair of vectors and a hermitian form, and a pair of matrices and a hermitian form has been fixed. A series of test files is now included in the tst directory. Alexander Konovalov pointed out the the init.g and read.g files had windows line breaks, this is also fixed. Finally, the documentation has been recompiled with the MathJax option.

Max Horn pointed out that we still used the deprecated GAP_ROOT_PATHS. This has been changed now into GAPInfo.RootPaths in version 1.2.4. More tests have been added to reach a better code coverage. Due to these tests, a bug in one of the methods for `EvaluateForm` was discovered and fixed. Alexander Konovalov noted that we used the deprecated `ReadTest` in our test files. This has been changed to `Test`. Finally some LaTeX issues were resolved in the documentation.

In verions 1.2.5, some small changes were made to the recognition part. Some new examples in the documentation explain better the functionality.

Version 1.2.6 is an intermediate update. It contains a number of corrections/additions suggested/implemented by Thomas Breuer and Max Horn, including an extension of some GAP library functions to create classical matrix groups. These additions are not yet documented.

In version 1.2.7, only an issue with the automatic release was fixed.

Version 1.2.8 optimizes the computation of base change matrices, making it faster by several orders of magnitude for large inputs. Moreover, Max Horn was added to the list of maintainers. Finally, various janitorial changes were made.

Chapter 2

Examples

Here we give some simple examples that display some of the functionality of Forms.

2.1 A conic of PG$(2,8)$

Consider the three-dimensional vector space V over the finite field GF(8) , and consider the following quadratic polynomial in 3 variables:

$$x_1^2 + x_2x_3.$$

Then this polynomial defines a quadratic form on V and the zeros form a *conic* of the associated projective plane. So in particular, our quadratic form defines a degenerate parabolic quadric of Witt Index 1. We will see now how we can use Forms to view this example.

Example

```
gap> gf := GF(8);
GF(2^3)
gap> vec := gf^3;
( GF(2^3)^3 )
gap> r := PolynomialRing( gf, 3);
PolynomialRing(..., [ x_1, x_2, x_3 ])
gap> poly := r.1^2 + r.2 * r.3;
x_1^2+x_2*x_3
gap> form := QuadraticFormByPolynomial( poly, r );
< quadratic form >
gap> Display( form );
Quadratic form
Gram Matrix:
 1 . .
 . . 1
 . . .
Polynomial: x_1^2+x_2*x_3
gap> IsDegenerateForm( form );
#I  Testing degeneracy of the *associated bilinear form*
true
gap> IsSingularForm( form );
false
gap> WittIndex( form );
1
gap> IsParabolicForm( form );
```

```
true
gap> RadicalOfForm( form );
<vector space over GF(2^3), with 0 generators>
```

Now our conic is stabilised by a group isomorphic to $\mathrm{GO}(3,8)$, but which is not identical to the group returned by the GAP command `GO(3,8)`. However, our conic is the canonical conic given in Forms.

Example

```
gap> canonical := IsometricCanonicalForm( form );
< parabolic quadratic form >
gap> form = canonical;
true
```

So we "change forms"...

Example

```
gap> go := GO(3,8);
GO(0,3,8)
gap> mat := InvariantQuadraticForm( go )!.matrix;
[ [ Z(2)^0, 0*Z(2), 0*Z(2) ], [ 0*Z(2), 0*Z(2), 0*Z(2) ],
  [ 0*Z(2), Z(2)^0, 0*Z(2) ] ]
gap> gapform := QuadraticFormByMatrix( mat, GF(8) );
< quadratic form >
gap> b := BaseChangeToCanonical( gapform );
[ [ Z(2)^0, 0*Z(2), 0*Z(2) ], [ 0*Z(2), Z(2)^0, 0*Z(2) ],
  [ 0*Z(2), 0*Z(2), Z(2)^0 ] ]
gap> hom := BaseChangeHomomorphism( b, GF(8) );
^[ [ Z(2)^0, 0*Z(2), 0*Z(2) ], [ 0*Z(2), Z(2)^0, 0*Z(2) ],
  [ 0*Z(2), 0*Z(2), Z(2)^0 ] ]
gap> newgo := Image(hom, go);
Group(
[ [ [ Z(2)^0, 0*Z(2), 0*Z(2) ], [ 0*Z(2), Z(2^3), 0*Z(2) ], [ 0*Z(2), 0*Z(2),
          Z(2^3)^6 ] ],
  [ [ Z(2)^0, 0*Z(2), 0*Z(2) ], [ Z(2)^0, Z(2)^0, Z(2)^0 ],
      [ 0*Z(2), Z(2)^0, 0*Z(2) ] ] ])
```

Now we look at the action of our new $\mathrm{GO}(3,8)$ on the conic.

Example

```
gap> conic := Filtered(vec, x -> IsZero( x^form ));;
gap> Size(conic);
64
gap> orbs := Orbits(newgo, conic, OnRight);;
gap> List(orbs,Size);
[ 1, 63 ]
```

So we see that there is a fixed point, which is actually the *nucleus* of the conic, or in other words, the radical of the form.

2.2 A form for $\mathrm{W}(5,3)$

The symplectic polar space $\mathrm{W}(5,q)$ is defined by an alternating reflexive bilinear form on the six-dimensional vector space over the finite field $\mathrm{GF}(q)$. Any invertible 6×6 matrix A which satisfies

$A+A^T=0$ is a candidate for the Gram matrix of a symplectic polarity. The canonical form we adopt in Forms for an alternating form is

$$f(x,y)=x_1y_2-x_2y_1+x_3y_4-x_4y_3\cdots+x_{2n-1}y_{2n}-x_{2n}y_{2n-1}.$$

Example

```
gap> f := GF(3);
GF(3)
gap> gram := [
> [0,0,0,1,0,0],
> [0,0,0,0,1,0],
> [0,0,0,0,0,1],
> [-1,0,0,0,0,0],
> [0,-1,0,0,0,0],
> [0,0,-1,0,0,0]] * One(f);;
gap> form := BilinearFormByMatrix( gram, f );
< bilinear form >
gap> IsSymplecticForm( form );
true
gap> Display( form );
Symplectic form
Gram Matrix:
 . . . 1 . .
 . . . . 1 .
 . . . . . 1
 2 . . . . .
 . 2 . . . .
 . . 2 . . .
gap> b := BaseChangeToCanonical( form );
[ [ Z(3)^0, 0*Z(3), 0*Z(3), 0*Z(3), 0*Z(3), 0*Z(3) ],
  [ 0*Z(3), 0*Z(3), 0*Z(3), Z(3)^0, 0*Z(3), 0*Z(3) ],
  [ 0*Z(3), Z(3)^0, 0*Z(3), 0*Z(3), 0*Z(3), 0*Z(3) ],
  [ 0*Z(3), 0*Z(3), 0*Z(3), 0*Z(3), Z(3)^0, 0*Z(3) ],
  [ 0*Z(3), 0*Z(3), Z(3)^0, 0*Z(3), 0*Z(3), 0*Z(3) ],
  [ 0*Z(3), 0*Z(3), 0*Z(3), 0*Z(3), 0*Z(3), Z(3)^0 ] ]
gap> Display( b );
 1 . . . . .
 . . . 1 . .
 . 1 . . . .
 . . . . 1 .
 . . 1 . . .
 . . . . . 1
gap> Display( b * gram * TransposedMat(b) );
 . 1 . . . .
 2 . . . . .
 . . . 1 . .
 . . 2 . . .
 . . . . . 1
 . . . . 2 .
```

2.3 What is the form preserved by this group?

Here we start with a matrix group which is available in GAP, namely $\mathrm{GO}(5,5)$. We then conjugate this group by an element of $\mathrm{GL}(5,5)$, and then we find the forms left invariant by this copy of $\mathrm{GO}(5,5)$ (which we expect to be a symmetric bilinear form).

Example

```
gap> go := GO(5, 5);
GO(0,5,5)
gap> x :=
> [ [ Z(5)^0, Z(5)^3, 0*Z(5), Z(5)^3, Z(5)^3 ],
>   [ Z(5)^2, Z(5)^3, 0*Z(5), Z(5)^2, Z(5) ],
>   [ Z(5)^2, Z(5)^2, Z(5)^0, Z(5), Z(5)^3 ],
>   [ Z(5)^0, Z(5)^3, Z(5), Z(5)^0, Z(5)^3 ],
>   [ Z(5)^3, 0*Z(5), Z(5)^0, 0*Z(5), Z(5) ]
>  ];;
gap> go2 := go^x;
<matrix group of size 18720000 with 2 generators>
gap> forms := PreservedSesquilinearForms( go2 );
[ < bilinear form > ]
gap> Display( forms[1] );
Bilinear form
Gram Matrix:
 4 2 4 3 3
 2 2 2 3 3
 4 2 3 1 4
 3 3 1 2 4
 3 3 4 4 3
```

Chapter 3

Background Theory on Forms

In this chapter we give a very brief overview of the theory of sesquilinear and quadratic forms. The reader can find more in the texts: Cameron [Cam00], Taylor [Tay92], Aschbacher [Asc00], or Kleidman and Liebeck [KL90].

3.1 Sesquilinear forms

A *sesquilinear form* on an n-dimensional vector space V over a field F, is a map f from $V \times V$ to F which is linear in the first coordinate, but semilinear in the second coordinate; that is, there is a field automorphism α (the *companion automorphism* of f) such that

$$f(v, \lambda w) = \lambda^{\alpha} f(v, w),$$

for all $v, w \in V$ and $\lambda \in F$. If α is the identity, then f is *bilinear*.

A bilinear form is *reflexive* if $f(v,w) = 0 \Rightarrow f(w,v) = 0$ for all $v, w \in V$. A bilinear form is *symmetric* if $f(v,w) = f(w,v)$ for all $v, w \in V$. It is clear that a symmetric bilinear form is reflexive. A bilinear form is *alternating* if $f(v,v) = 0$ for all $v \in V$. Using the linearity to compute $f(v+w, v+w)$, we see that an alternating form is also reflexive. When the characteristic of the field differs from 2, an alternating form f can also be characterised as $f(v,w) = -f(w,v)$ for all $v, w \in V$. It can be proved (see Chapter 7 of [Tay92]) that symmetric and alternating bilinear forms are the only reflexive bilinear forms.

For a given sesquilinear form f, the choice of the basis determines uniquely an $n \times n$ matrix M such that $f(v,w) = vMw^{\alpha T}$.

This matrix is also called the *Gram matrix* of f. Given a sesquilinear form f, we will denote its Gram matrix by M_f. In Forms, sesquilinear forms can be constructed using matrices or polynomials, where we always suppose that the basis of the vector space is the standard basis (i.e., the rows of the identity matrix).

One proves easily that a bilinear form f is symmetric if and only if M_f is a symmetric matrix, i.e., $M_f^T = M_f$, and that a bilinear form f is alternating if and only if M_f is a skew symmetric matrix, i.e., $M_f^T = -M_f$. When the characteristic of the field is two, the condition that $f(v,v) = 0$ for all $v \in V$ implies $M_f^T = M_f$ AND $(M_{ii}) = 0$ for all i (and where the matrix $M_f = (M_{ij})$). Since any skew symmetric odd dimensional matrix is singular, it follows that an alternating form of an odd dimensional vector space is degenerate.

A sesquilinear form f is *hermitian* (n.b., *conjugate-symmetric* in [CCN+85]) if $f(v,w) = f(w,v)^{\alpha}$ holds for all vectors v, w, with α an involutory field automorphism only dependent on f. Again, it

can be easily proved that a sesquilinear form f is hermitian if and only if $M_f^T = M_f^\alpha$ (i.e., a hermitian matix). It is proved (see Chapter 7 of [Tay92]) that hermitian forms are the only reflexive sesquilinear forms that are not bilinear. Hence, in general, all reflexive sesquilinear forms are known, they are either hermitian or bilinear, and in the latter case, they are either symmetric or alternating (again, see Chapter 7 of [Tay92]).

In Forms, only the construction of REFLEXIVE sesquilinear forms is allowed. An error message will be displayed if any attempt to construct a non-reflexive sesquilinear form is made. As a consequence, the Gram Matrix of a sesquilinear form is always a symmetric, a skew symmetric or a hermitian matrix. From now on, the notion of a "sesquilinear form" will always refer to a "reflexive sesquilinear form".

Given a sesquilinear form f, two vectors v and w are *orthogonal* with respect to f if $f(v,w) = 0$. Note that the reflexivity makes orthogonality between two vectors a symmetric relation. A vector v is called *isotropic* if $f(v,v) = 0$. The *radical* of f (n.b., *kernel* in [CCN+85]) is the subspace consisting of vectors which are orthogonal to every vector. That is,

$$\mathrm{Rad}(f) = \{v \in V | f(v,w) = 0, \forall w \in V\},$$

and we say that f is *non-degenerate* if its radical is trivial (and *degenerate* otherwise).

Given a subspace W, we denote the set of vectors of V orthogonal with all vectors of W by $W^\perp$. We call a subspace W *totally isotropic* with respect to f if W is contained in $W^\perp$, i.e.

$$f(v,w) = 0, \forall v,w \in W.$$

Suppose that f is a non-degenerate sesquilinear form. The *Witt index* of f is the maximum dimension of a totally isotropic subspace with respect to f.

Let f be a sesquilinear form on $V(n,q)$, with radical R, a k-dimensional subspace of $V(n,q)$, $0 \leq k \leq n$. Then f induces a non-degenerate form g on V/R. When $dim(R) = 0$, then $g = f$ and f is non-degenerate. Notice that all totally isotropic subspaces of maximal dimension of a degenerate form f contain the radical of f. In Forms, the notion Witt index will ALWAYS REFER TO THE INDUCED NON-DEGENERATE FORM g. Hence, given a degenerate form f, computing its Witt index will return the Witt index of the induced form g. THIS ALSO HOLDS FOR THE NOTIONS ELLIPTIC, PARABOLIC AND HYPERBOLIC FOR A BILINEAR FORM, WHICH ARE NOTIONS DEFINED USING THE WITT INDEX, SEE BELOW.

We end this section with a short description of the conventions used in Forms for the notions orthogonal, symplectic, pseudo, hyperbolic, elliptic and parabolic. We call a form f *symplectic* if and only if f is alternating. When the characteristic of the field is odd, we call a form f *orthogonal* if and only f is symmetric, and when the characteristic of the field is even, we call a form f *pseudo* if and only if f is symmetric but not alternating. This terminology is related to the theory of polar spaces, and in the case of orthogonal forms, we adopt the terms *hyperbolic*, *elliptic* and *parabolic* for the three different isometry types of orthogonal forms. From the point of view of matrix groups, these three types correspond as follows. Recall that, as explained above, the Witt index refers to the Witt index of the INDUCED NON-DEGENERATE FORM g when f is degenerate.

Hyperbolic	Orthogonal of + type	$V/\mathrm{Rad}(f)$ has even dimension, g has maximal Witt index
Elliptic	Orthogonal of - type	$V/\mathrm{Rad}(f)$ has even dimension, g has non-maximal Witt index
Parabolic	Orthogonal of o type	$V/\mathrm{Rad}(f)$ has odd dimension

Table: Posibilites for an orthogonal form f on a vector space V

3.1.1 Examples

The examples we present in this section do not demonstrate the entire suite of operations entailed in Forms. They are intended to allow the user to become familiar with particular aspects of this package. All the functionality for sesquilinear forms will be listed in detail in the next chapter.

We try to construct a bilinear form...

Example

```
gap> mat := [[1,0,0],[0,1,4],[1,2,1]]*Z(5)^0;
[ [ Z(5)^0, 0*Z(5), 0*Z(5) ], [ 0*Z(5), Z(5)^0, Z(5)^2 ],
  [ Z(5)^0, Z(5), Z(5)^0 ] ]
gap> form := BilinearFormByMatrix(mat,GF(5));
Error, Invalid Gram matrix
 called from
BilinearFormByMatrixOp( MutableCopyMat( m ), f
 ) at ./pkg/forms/lib/forms.gi:164 called from
<function "unknown">( <arguments> )
 called from read-eval loop at line 8 of *stdin*
you can 'quit;' to quit to outer loop, or
you can 'return;' to continue
brk> quit;
```

It is clear that the matrix used is not defining a reflexive bilinear form, which causes the system to generate the error message.

We construct now a reflexive bilinear form. We investigate also the radical of the form.

Example

```
gap> mat := [[1,0,0,0],[0,1,0,0],[0,0,1,0],[0,0,0,-1]]*Z(9)^0;
[ [ Z(3)^0, 0*Z(3), 0*Z(3), 0*Z(3) ], [ 0*Z(3), Z(3)^0, 0*Z(3), 0*Z(3) ],
  [ 0*Z(3), 0*Z(3), Z(3)^0, 0*Z(3) ], [ 0*Z(3), 0*Z(3), 0*Z(3), Z(3) ] ]
gap> form := BilinearFormByMatrix(mat,GF(9));
< bilinear form >
gap> Display(form);
Bilinear form
Gram Matrix:
 1 . . .
 . 1 . .
 . . 1 .
 . . . 2
gap> IsReflexiveForm(form);
true
gap> IsSymmetricForm(form);
true
gap> IsAlternatingForm(form);
false
gap> r := RadicalOfForm(form);
<vector space over GF(3^2), with 0 generators>
gap> Dimension(r);
0
```

Degenerate forms are allowed. As an example we construct an alternating bilinear form on an odd dimensional vector space.

Example

```
gap> mat := [[0,0,-2],[0,0,1],[2,-1,0]]*Z(7)^0;
[ [ 0*Z(7), 0*Z(7), Z(7)^5 ], [ 0*Z(7), 0*Z(7), Z(7)^0 ],
```

```
  [ Z(7)^2, Z(7)^3, 0*Z(7) ] ]
gap> form := BilinearFormByMatrix(mat,GF(7));
< bilinear form >
gap> Display(form);
Bilinear form
Gram Matrix:
 . . 5
 . . 1
 2 6 .
gap> IsSymmetricForm(form);
false
gap> IsAlternatingForm(form);
true
gap> r := RadicalOfForm(form);
<vector space over GF(7), with 1 generators>
gap> Dimension(r);
1
```

When the characteristic of the field equals two, alternating forms are also symmetric. We construct an example.

Example

```
gap> mat := [[0,1,0,0,0,0],[1,0,0,0,0,0],[0,0,0,0,0,1],
>         [0,0,0,0,1,0],[0,0,0,1,0,0],[0,0,1,0,0,0]]*Z(16)^0;
[ [ 0*Z(2), Z(2)^0, 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2) ],
  [ Z(2)^0, 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2) ],
  [ 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2), Z(2)^0 ],
  [ 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2), Z(2)^0, 0*Z(2) ],
  [ 0*Z(2), 0*Z(2), 0*Z(2), Z(2)^0, 0*Z(2), 0*Z(2) ],
  [ 0*Z(2), 0*Z(2), Z(2)^0, 0*Z(2), 0*Z(2), 0*Z(2) ] ]
gap> form := BilinearFormByMatrix(mat,GF(16));
< bilinear form >
gap> Display(form);
Bilinear form
Gram Matrix:
 . 1 . . . .
 1 . . . . .
 . . . . . 1
 . . . . 1 .
 . . . 1 . .
 . . 1 . . .
gap> IsSymmetricForm(form);
true
gap> IsAlternatingForm(form);
true
gap> IsDegenerateForm(form);
false
gap> WittIndex(form);
3
```

To define a hermitian form, we need a matrix and the companion automorphism. Since this automorphism has order 2, it exists and is unique if the ground field has square order. In the next example, the

chosen matrix is somewhat special. Together with the companion automorphism, it determines a hermitian sesquilinear form. Without the companion automorphism, it determines an alternating bilinear form.

Example

```
gap> mat := [[0*Z(5),0*Z(5),0*Z(25),Z(25)^3],[0*Z(5),0*Z(5),Z(25)^3,0*Z(25)],
>         [0*Z(5),-Z(25)^3,0*Z(5),0*Z(5)],[-Z(25)^3,0*Z(5),0*Z(25),0*Z(25)]];
[ [ 0*Z(5), 0*Z(5), 0*Z(5), Z(5^2)^3 ], [ 0*Z(5), 0*Z(5), Z(5^2)^3, 0*Z(5) ],
  [ 0*Z(5), Z(5^2)^15, 0*Z(5), 0*Z(5) ],
  [ Z(5^2)^15, 0*Z(5), 0*Z(5), 0*Z(5) ] ]
gap> form := HermitianFormByMatrix(mat,GF(25));
< hermitian form >
gap> Display(form);
Hermitian form
Gram Matrix:
z = Z(25)
    .    .    .  z^3
    .    .  z^3    .
    . z^15    .    .
 z^15    .    .    .
gap> WittIndex(form);
2
gap> form2 := BilinearFormByMatrix(mat,GF(25));
< bilinear form >
gap> Display(form2);
Bilinear form
Gram Matrix:
z = Z(25)
    .    .    .  z^3
    .    .  z^3    .
    . z^15    .    .
 z^15    .    .    .
gap> IsAlternatingForm(form2);
true
gap> Display(IsometricCanonicalForm(form));
Hermitian form
Gram Matrix:
 1 . . .
 . 1 . .
 . . 1 .
 . . . 1
Witt Index: 2
gap> Display(IsometricCanonicalForm(form2));
Bilinear form
Gram Matrix:
 . 1 . .
 4 . . .
 . . . 1
 . . 4 .
Witt Index: 2
```

We continue the previous example by exploring a little bit the sesquilinear form *form*, and hence demonstrate some of the functionality of the Forms package. Eventually, we find a 2-dimensional

totally isotropic subspace, which lets us conclude that the Witt index of *form* is at least 2, which is confirmed afterwards by calling the appropriate function.

Example

```
gap> V := GF(25)^4;
( GF(5^2)^4 )
gap> u := [Z(5)^0,Z(5^2)^11,Z(5)^3,Z(5^2)^13 ];
[ Z(5)^0, Z(5^2)^11, Z(5)^3, Z(5^2)^13 ]
gap> [u,u]^form;
0*Z(5)
gap> v := [Z(5)^0,Z(5^2)^5,Z(5^2),Z(5^2)^13 ];
[ Z(5)^0, Z(5^2)^5, Z(5^2), Z(5^2)^13 ]
gap> [v,v]^form;
0*Z(5)
gap> [u,v]^form;
Z(5^2)^7
gap> ([v,u]^form)^5;
Z(5^2)^7
gap> w := [Z(5^2)^21,Z(5^2)^19,Z(5^2)^4,Z(5)^3 ];
[ Z(5^2)^21, Z(5^2)^19, Z(5^2)^4, Z(5)^3 ]
gap> [w,w]^form;
Z(5)
gap> v := [Z(5)^0,Z(5^2)^10,Z(5^2)^15,Z(5^2)^3 ];
[ Z(5)^0, Z(5^2)^10, Z(5^2)^15, Z(5^2)^3 ]
gap> u := [Z(5)^3,Z(5^2)^9,Z(5^2)^4,Z(5^2)^16 ];
[ Z(5)^3, Z(5^2)^9, Z(5^2)^4, Z(5^2)^16 ]
gap> w := [Z(5)^2,Z(5^2)^9,Z(5^2)^23,Z(5^2)^11 ];
[ Z(5)^2, Z(5^2)^9, Z(5^2)^23, Z(5^2)^11 ]
gap> [u,v]^form;
0*Z(5)
gap> [u,w]^form;
0*Z(5)
gap> [v,w]^form;
0*Z(5)
gap> s := Subspace(V,[v,u,w]);
<vector space over GF(5^2), with 3 generators>
gap> Dimension(s);
2
gap> WittIndex(form);
2
```

3.2 Quadratic forms

A *quadratic form* on an n-dimensional vector space V over a field F, is a map Q from V to F satisfying the following two conditions:

$$Q(\lambda v) = \lambda^2 Q(v), \forall \lambda \in F, \forall v \in V,$$

and, the map f defined from $V \times V$ to F as follows,

$$f(v,w) := Q(v+w) - Q(v) - Q(w),$$

is a bilinear form on V. From this definition it follows that $f(v,v) = Q(2v) - 2Q(v) = 2Q(v)$.

The associated bilinear form f (which is called the *polar form* of Q in [CCN+85]) is clearly reflexive. When the characteristic of the field is odd, it is clear that f is a symmetric bilinear form. The equation $f(v,v) = 2Q(v)$ allows us to reconstruct the quadratic form from the bilinear form, and hence there is a one-to-one correspondence between quadratic forms and symmetric bilinear forms. When the characteristic of the field equals 2, the bilinear form f is alternating (from the fact that $f(v,v) = 2Q(v) = 0$). Note, however, that different quadratic forms can determine the same alternating form.

As in the case of sesquilinear forms, we will associate a matrix to a quadratic form. Chosing a basis of the vector space V, it is clear that an $n \times n$ matrix determines the quadratic form completely. In Forms, the *Gram matrix* of a quadratic form is always an upper triangle matrix M, such that

$$Q(v) = vMv^T,$$

where the basis of V is the standard basis. Although the Gram matrix stored with the quadratic form is always an upper triangle matrix, the user is allowed to use any matrix to define the quadratic form, since any matrix M defines a quadratic form $Q(v) := vMv^T$. During the construction, an appopriate upper triangle matrix is computed and stored as the Gram matrix. So the Gram matrix of the associated bilinear form is $M + M^T$.

The associated bilinear form could be used to define the notions "isotropic", "totally isotropic" and "non-degenerate", however, under these restrictions the geometry of quadratic forms in even characteristic is lost. In most of the literature, these notions refer indeed to the associated bilinear form, and the notion of "singularity" is added to regain the geometrical structure.

In Forms, we use the above described approach. This means that a vector is isotropic if and only if it is isotropic with respect to the associated bilinear form. A subspace is totally isotropic if and only if it is totally isotropic with respect to the associated bilinear form, and we call the quadratic form degenerate if and only if the associated bilinear form is degenerate.

A vector v is called *singular* with relation to the quadratic form Q if and only if $Q(v) = 0$. two vectors v and w are *orthogonal* with respect to Q if and only if they are orthogonal with respect to the associated bilinear form f. The *radical* of the quadratic form Q, is the intersection of the set of all singular vectors with relation to Q and the radical of the associated bilinear form f, i.e.

$$\mathrm{Rad}(Q) = \{v \in V | Q(v) = 0 \text{ and } v \in \mathrm{Rad}(f)\}.$$

We call a quadratic form Q *non-singular* if and only if the radical contains only the zero vector, and *singular* otherwise.

A subspace W of the vector space is called *totally singular* if and only if all vectors of W are singular, i.e., Q vanishes totally on W. Necessarily, a totally singular subspace is also totally isotropic with relation to the associated bilinear form f, but the converse is only true when the characteristic of the field is odd.

Suppose now that Q is a non-singular quadratic form. The *Witt index* of Q is the maximum dimension of a totally singular subspace with respect to Q.

Let Q be a quadratic form on $V(n,q)$, with radical R, a k-dimensional subspace of $V(n,q)$, $0 \leq k \leq n$. Then Q induces a non-singular form Q' on V/R. When $dim(R) = 0$, then $Q = Q'$ and Q is non-singular. Notice that all totally singular subspaces of maximal dimension of a singular form Q contain the radical of Q. In Forms, the notion Witt index will ALWAYS REFER TO THE INDUCED NON-SINGULAR FORM Q'. Hence, given a singular form Q, computing its Witt index will return the Witt index of the induced form Q'. THIS ALSO HOLDS FOR THE NOTIONS ELLIPTIC, PARABOLIC

AND HYPERBOLIC FOR A QUADRATIC FORM, WHICH ARE NOTIONS DEFINED USING THE WITT INDEX, SEE BELOW.

The terminology*hyperbolic*, *elliptic* and *parabolic* is also used for quadratic forms, and is defined analogously as for the bilinear forms using the Witt index. Also in the case of quadratic forms, this terminology is related to the theory of polar spaces. Recall that, as explained above, the Witt index refers to the Witt index of the INDUCED NON-SINGULAR FORM Q when Q' is singular.

Hyperbolic	Orthogonal of + type	$V/\mathrm{Rad}(Q)$ has even dimension, Q' has maximal Witt index
Elliptic	Orthogonal of - type	$V/\mathrm{Rad}(Q)$ has even dimension, Q' has non-maximal Witt index
Parabolic	Orthogonal of o type	$V/\mathrm{Rad}(Q)$ has odd dimension

Table: Posibilites for a quadratic form Q on a vector space V

From the above definitions, it follows that, when the characteristic of the field differs from 2, a quadratic form Q is non-singular if and only if its associated bilinear form f is non-degenerate. When the characteristic of the field is 2, one can easily construct non-singular quadratic forms, with a degenerate associated bilinear form. We will give an example of this situation in the next section.

3.2.1 Examples

We construct some quadratic forms to demonstrate some funcionality of Forms. As in the previous example section, they are intended to allow the user to gain some familiarity. All the functionality for quadratic forms will be listed in detail in the next chapter.

The user can construct quadratic forms using any matrix (provided it has the right dimension). The Gram matrix is always stored as an upper triangle matrix, as explained above.

Example

```
gap> V := GF(4)^3;
( GF(2^2)^3 )
gap> mat := [[Z(2^2)^2,Z(2^2),Z(2^2)^2],[Z(2^2)^2,Z(2)^0,Z(2)^0],
>         [0*Z(2),Z(2)^0,0*Z(2)]];
[ [ Z(2^2)^2, Z(2^2), Z(2^2)^2 ], [ Z(2^2)^2, Z(2)^0, Z(2)^0 ],
  [ 0*Z(2), Z(2)^0, 0*Z(2) ] ]
gap> qform := QuadraticFormByMatrix(mat, GF(4));
< quadratic form >
gap> Display( qform );
Quadratic form
Gram Matrix:
z = Z(4)
 z^2   1 z^2
   .   1   .
   .   .   .
gap> PolynomialOfForm( qform );
Z(2^2)^2*x_1^2+x_1*x_2+Z(2^2)^2*x_1*x_3+x_2^2
```

In the previous example, we saw how we used a polynomial to display a quadratic form. Conversely, Forms allows the user to construct (quadratic) forms using a polynomial.

Example

```
gap> r := PolynomialRing(GF(8),4);
GF(2^3)[x_1,x_2,x_3,x_4]
gap> poly := r.1*r.2+r.3*r.4;
```

```
x_1*x_2+x_3*x_4
gap> qform := QuadraticFormByPolynomial(poly, r);
< quadratic form >
gap> Display(qform);
Quadratic form
Gram Matrix:
 . 1 . .
 . . . .
 . . . 1
 . . . .
Polynomial: x_1*x_2+x_3*x_4
gap> RadicalOfForm(qform);
<vector space over GF(2^3), with 0 generators>
```

We construct now two different quadratic forms with the same associated bilinear form.

Example

```
gap> mat := [[Z(16)^3,1,0,0],[0,Z(16)^5,0,0],
>             [0,0,Z(16)^3,1],[0,0,0,Z(16)^12]]*Z(16)^0;
[ [ Z(2^4)^3, Z(2)^0, 0*Z(2), 0*Z(2) ], [ 0*Z(2), Z(2^2), 0*Z(2), 0*Z(2) ],
  [ 0*Z(2), 0*Z(2), Z(2^4)^3, Z(2)^0 ], [ 0*Z(2), 0*Z(2), 0*Z(2), Z(2^4)^12 ]
 ]
gap> qform := QuadraticFormByMatrix(mat,GF(16));
< quadratic form >
gap> Display( qform );
Quadratic form
Gram Matrix:
z = Z(16)
  z^3    1    .    .
    .  z^5    .    .
    .    .  z^3    1
    .    .    . z^12
gap> mat2 := [[Z(16)^7,1,0,0],[0,0,0,0],
>             [0,0,Z(16)^2,1],[0,0,0,Z(16)^9]]*Z(16)^0;
[ [ Z(2^4)^7, Z(2)^0, 0*Z(2), 0*Z(2) ], [ 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2) ],
  [ 0*Z(2), 0*Z(2), Z(2^4)^2, Z(2)^0 ], [ 0*Z(2), 0*Z(2), 0*Z(2), Z(2^4)^9 ] ]
gap> qform2 := QuadraticFormByMatrix(mat2, GF(16));
< quadratic form >
gap> Display( qform2 );
Quadratic form
Gram Matrix:
z = Z(16)
  z^7    1    .    .
    .    .    .    .
    .    .  z^2    1
    .    .    .  z^9
gap> biform := AssociatedBilinearForm( qform2 );
< bilinear form >
gap> Display( biform );
Bilinear form
Gram Matrix:
 . 1 . .
 1 . . .
```

```
. . . 1
. . 1 .
```

We end with an example of a non-singular quadratic form with a degenerate associated bilinear form.

Example

```
gap> mat := [ [ Z(2^2), Z(2^2), Z(2^2), Z(2^2), Z(2^2) ],
>    [ 0*Z(2), Z(2^2), Z(2^2)^2, 0*Z(2), Z(2)^0 ],
>    [ 0*Z(2), 0*Z(2), Z(2)^0, Z(2)^0, Z(2)^0 ],
>    [ 0*Z(2), 0*Z(2), 0*Z(2), Z(2)^0, Z(2)^0 ],
>    [ 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2), Z(2)^0 ] ];;
gap> qform := QuadraticFormByMatrix(mat,GF(4));
< quadratic form >
gap> IsSingularForm(qform);
false
gap> IsDegenerateForm(qform);
#I  Testing degeneracy of the *associated bilinear form*
true
gap> biform := AssociatedBilinearForm(qform);
< bilinear form >
gap> Display(biform);
Bilinear form
Gram Matrix:
z = Z(4)
   . z^1 z^1 z^1 z^1
 z^1   . z^2   .   1
 z^1 z^2   .   1   1
 z^1   .   1   .   1
 z^1   1   1   1   .
gap> IsDegenerateForm(biform);
true
```

Chapter 4

Constructing forms and basic functionality

In this chapter, all operations to construct sesquilinear and quadratic forms are listed, along with their basic attributes and properties.

4.1 Important filters

4.1.1 Categories for forms

▷ `IsBilinearForm` (Category)
▷ `IsHermitianForm` (Category)
▷ `IsSesquilinearForm` (Category)
▷ `IsQuadraticForm` (Category)
▷ `IsForm` (Category)
▷ `IsTrivialForm` (Category)

The categories `IsBilinearForm` and `IsHermitianForm` are categories for bilinear and hermitian forms, respectively. They are disjoint and are both contained in the category `IsSesquilinearForm`.

Quadratic forms are contained in the category `IsQuadraticForm`. The categories `IsSesquilinearForm` and `IsQuadraticForm` are disjoint and are both contained in the category `IsForm`.

The user is allowed to construct the trivial form (mapping all vectors to the zero element of the field). The trivial form is an object in the category `IsTrivialForm`. This category is contained in `IsForm` and disjoint from `IsSesquilinearForm` and `IsQuadraticForm`.

4.1.2 Representation for forms

▷ `IsFormRep` (Representation)

Every form is represented by a matrix, the base field and a string describing the "type" of the form.

4.2 Constructing forms using a matrix

4.2.1 BilinearFormByMatrix

▷ `BilinearFormByMatrix(`*`matrix[, field]`*`)` (operation)

Returns: a bilinear form

The argument *`matrix`* must be a symmetric, or skew-symmetric, square matrix over the finite field *`field`*. The argument *`field`* is an optional argument, and if it is not given, then we assume that the *defining field* of the bilinear form is the smallest field containing the entries of matrix. Below we give an example where the defining field can make a difference in some applications. As it is only possible to construct reflexive bilinear forms, it is checked whether the matrix *`matrix`* is symmetric or skew symmetric. If matrix *`matrix`* is not symmetric nor skew symmetric, then an error message is returned. The output is a bilinear form (i.e., an object in `IsBilinearForm`) with Gram matrix *`matrix`* and defining field *`field`*. (See 3.1 for more on bilinear forms).

Example

```
gap> mat := IdentityMat(4, GF(9));
[ [ Z(3)^0, 0*Z(3), 0*Z(3), 0*Z(3) ], [ 0*Z(3), Z(3)^0, 0*Z(3), 0*Z(3) ],
  [ 0*Z(3), 0*Z(3), Z(3)^0, 0*Z(3) ], [ 0*Z(3), 0*Z(3), 0*Z(3), Z(3)^0 ] ]
gap> form := BilinearFormByMatrix(mat,GF(9));
< bilinear form >
gap> Display(form);
Bilinear form
Gram Matrix:
 1 . . .
 . 1 . .
 . . 1 .
 . . . 1
gap> mat := [[0*Z(2),Z(16)^12,0*Z(2),Z(4)^2,Z(16)^13],
>    [Z(16)^12,0*Z(2),0*Z(2),Z(16)^11,Z(16)],
>    [0*Z(2),0*Z(2),0*Z(2),Z(4)^2,Z(16)^3],
>    [Z(4)^2,Z(16)^11,Z(4)^2,0*Z(2),Z(16)^3],
>    [Z(16)^13,Z(16),Z(16)^3,Z(16)^3,0*Z(2) ]];
[ [ 0*Z(2), Z(2^4)^12, 0*Z(2), Z(2^2)^2, Z(2^4)^13 ],
  [ Z(2^4)^12, 0*Z(2), 0*Z(2), Z(2^4)^11, Z(2^4) ],
  [ 0*Z(2), 0*Z(2), 0*Z(2), Z(2^2)^2, Z(2^4)^3 ],
  [ Z(2^2)^2, Z(2^4)^11, Z(2^2)^2, 0*Z(2), Z(2^4)^3 ],
  [ Z(2^4)^13, Z(2^4), Z(2^4)^3, Z(2^4)^3, 0*Z(2) ] ]
gap> form := BilinearFormByMatrix(mat,GF(16));
< bilinear form >
gap> Display(form);
Bilinear form
Gram Matrix:
z = Z(16)
    . z^12    . z^10 z^13
 z^12    .    . z^11  z^1
    .    .    . z^10  z^3
 z^10 z^11 z^10    .  z^3
 z^13  z^1  z^3  z^3    .
gap> mat := [[1,0,0,0],[0,1,0,0],[0,0,0,1],[0,0,1,0]]*Z(7)^0;
[ [ Z(7)^0, 0*Z(7), 0*Z(7), 0*Z(7) ], [ 0*Z(7), Z(7)^0, 0*Z(7), 0*Z(7) ],
  [ 0*Z(7), 0*Z(7), 0*Z(7), Z(7)^0 ], [ 0*Z(7), 0*Z(7), Z(7)^0, 0*Z(7) ] ]
gap> form := BilinearFormByMatrix(mat);
< bilinear form >
gap> WittIndex(form);
```

```
1
gap> form := BilinearFormByMatrix(mat,GF(49));
< bilinear form >
gap> WittIndex(form);
2
```

4.2.2 QuadraticFormByMatrix

▷ QuadraticFormByMatrix(*matrix[, field]*) (operation)

Returns: a quadratic form

The argument *matrix* must be a square matrix over the finite field *field*. The argument *field* is an optional argument, and if it is not given, then we assume that the *defining field* of the bilinear form is the smallest field containing the entries of matrix. Below we give an example where the defining field can make a difference in some applications. Any square matrix determines a quadratic form, but the Gram matrix is recomputed so that it is an upper triangle matrix. The output is a quadratic form (i.e., an object in IsQuadraticForm) with defining field *field*. (See 3.2 for more on bilinear forms).

Example

```
gap> mat := [[1,0,0,0],[0,3,0,0],[0,0,0,6],[0,0,6,0]]*Z(7)^0;
[ [ Z(7)^0, 0*Z(7), 0*Z(7), 0*Z(7) ], [ 0*Z(7), Z(7), 0*Z(7), 0*Z(7) ],
  [ 0*Z(7), 0*Z(7), 0*Z(7), Z(7)^3 ], [ 0*Z(7), 0*Z(7), Z(7)^3, 0*Z(7) ] ]
gap> form := QuadraticFormByMatrix(mat,GF(7));
< quadratic form >
gap> Display(form);
Quadratic form
Gram Matrix:
 1 . . .
 . 3 . .
 . . . 5
 . . . .
gap> gf := GF(2^2);
GF(2^2)
gap> mat := InvariantQuadraticForm( SO(-1, 4, 4) )!.matrix;
[ [ 0*Z(2), Z(2)^0, 0*Z(2), 0*Z(2) ], [ 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2) ],
  [ 0*Z(2), 0*Z(2), Z(2^2)^2, Z(2)^0 ], [ 0*Z(2), 0*Z(2), 0*Z(2), Z(2^2)^2 ] ]
gap> form := QuadraticFormByMatrix( mat, gf );
< quadratic form >
gap> Display(form);
Quadratic form
Gram Matrix:
z = Z(4)
   .   1   .   .
   .   .   .   .
   .   . z^2   1
   .   .   . z^2
```

The following example shows how using the argument *field* has influence on the properties of the constructed form.

Example

```
gap> mat :=
> [[Z(2)^0,Z(2)^0,0*Z(2),0*Z(2)],[0*Z(2),Z(2)^0,0*Z(2),0*Z(2)],
>  [0*Z(2),0*Z(2),0*Z(2),Z(2)^0],[0*Z(2),0*Z(2),0*Z(2),0*Z(2)]];
```

```
[ [ Z(2)^0, Z(2)^0, 0*Z(2), 0*Z(2) ], [ 0*Z(2), Z(2)^0, 0*Z(2), 0*Z(2) ],
  [ 0*Z(2), 0*Z(2), 0*Z(2), Z(2)^0 ], [ 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2) ] ]
gap> form := QuadraticFormByMatrix(mat);
< quadratic form >
gap> WittIndex(form);
1
gap> form := QuadraticFormByMatrix(mat,GF(4));
< quadratic form >
gap> WittIndex(form);
2
```

4.2.3 HermitianFormByMatrix

▷ HermitianFormByMatrix(*matrix, field*) (operation)

Returns: a hermitian sesquilinear form

The argument *matrix* must be a hermitian square matrix over the finite field *field*, and *field* has square order. The field must be specified, since we can only determine the smallest field containing the entries of *matrix*. As it is only possible to construct reflexive sesquilinear forms, it is checked whether the matrix is a hermitian matrix, and if not, an error message is returned. The output is a hermitian sesquilinear form (i.e., an object in IsHermitianForm) with Gram matrix *matrix* and defining field *field*. (See 3.1 for more on hermitian forms).

Example

```
gap> gf := GF(3^2);
GF(3^2)
gap> mat := IdentityMat(4, gf);
[ [ Z(3)^0, 0*Z(3), 0*Z(3), 0*Z(3) ], [ 0*Z(3), Z(3)^0, 0*Z(3), 0*Z(3) ],
  [ 0*Z(3), 0*Z(3), Z(3)^0, 0*Z(3) ], [ 0*Z(3), 0*Z(3), 0*Z(3), Z(3)^0 ] ]
gap> form := HermitianFormByMatrix( mat, gf );
< hermitian form >
gap> Display(form);
Hermitian form
Gram Matrix:
 1 . . .
 . 1 . .
 . . 1 .
 . . . 1
gap> mat := [[Z(11)^0,0*Z(11),0*Z(11)],[0*Z(11),0*Z(11),Z(11)],
>     [0*Z(11),Z(11),0*Z(11)]];
[ [ Z(11)^0, 0*Z(11), 0*Z(11) ], [ 0*Z(11), 0*Z(11), Z(11) ],
  [ 0*Z(11), Z(11), 0*Z(11) ] ]
gap> form := HermitianFormByMatrix(mat,GF(121));
< hermitian form >
gap> Display(form);
Hermitian form
Gram Matrix:
  1  .  .
  .  .  2
  .  2  .
```

4.3 Constructing forms using a polynomial

Suppose that f is a sesquilinear form on an n-dimensional vectorspace. Consider a vector x with coordinates $x_1,\ldots,x_n$ with x_i indeterminates over the field. Then $f(x,x)$ is a polynomial in n indeterminates. When f is alternating, $f(x,x)$ is identically zero, but in all other cases, $f(x,x)$ determines f completely.

Conversely, suppose that Q is a quadratic form on an n-dimensional vectorspace. Consider a vector x with coordinates $x_1,\ldots,x_n$ with x_i indeterminates over the field. Then $Q(x)$ is a polynomial in n indeterminates, and $Q(x)$ determines Q completely.

Forms provides functionality to construct bilinear, hermitian and quadratic forms using an appropriate polynomial.

4.3.1 BilinearFormByPolynomial

▷ `BilinearFormByPolynomial(`*`poly, r[, n]`*`)` (operation)

Returns: a bilinear form

The argument *`poly`* must be a polynomial in the polynomial ring `r`. The (optional) last argument is the dimension for the underlying vector space of the resulting form, which by default is the number of indeterminates specified by *`poly`*. It is checked whether the polynomial is a homogeneous polynomial of degree two over the given field, and if not, an error message is returned. It is not possible to construct a nontrivial bilinear form from a polynomial in even characteristic. The output is a bilinear (orthogonal) form in the category `IsBilinearForm`. (See 3.1 for more on bilinear forms).

Example

```
gap> r := PolynomialRing( GF(11), 4);
GF(11)[x_1,x_2,x_3,x_4]
gap> vars := IndeterminatesOfPolynomialRing( r );
[ x_1, x_2, x_3, x_4 ]
gap> pol := vars[1]*vars[2]+vars[3]*vars[4];
x_1*x_2+x_3*x_4
gap> form := BilinearFormByPolynomial(pol, r, 4);
< bilinear form >
gap> Display(form);
Bilinear form
Gram Matrix:
  .  6  .  .
  6  .  .  .
  .  .  .  6
  .  .  6  .
Polynomial: x_1*x_2+x_3*x_4
gap> r := PolynomialRing(GF(4),2);
GF(2^2)[x_1,x_2]
gap> pol := r.1*r.2;
x_1*x_2
gap> form := BilinearFormByPolynomial(pol,r);
Error, No orthogonal form can be associated with a quadratic polynomial in even cha\
ra
cteristic
 called from
BilinearFormByPolynomial( pol, pring, n
 ) at ./pkg/forms/lib/forms.gi:470 called from
```

```
<function "unknown">( <arguments> )
 called from read-eval loop at line 14 of *stdin*
you can 'quit;' to quit to outer loop, or
you can 'return;' to continue
brk> quit;
```

4.3.2 QuadraticFormByPolynomial

▷ `QuadraticFormByPolynomial(`*poly, r[, n]*`)` (operation)

Returns: a quadratic form

The argument *poly* must be a polynomial in the polynomial ring *r*. The (optional) last argument is the dimension for the underlying vector space of the resulting form, which by default is the number of indeterminates specified by *poly*. It is checked whether the polynomial is a homogeneous polynomial of degree two over the given field, and if not, an error message is returned. The output is a quadratic form in the category `IsQuadraticForm`. (See 3.2 for more on quadratic forms).

Example

```
gap> r := PolynomialRing( GF(8), 3);
GF(2^3)[x_1,x_2,x_3]
gap> poly := r.1^2 + r.2^2 + r.3^2;
x_1^2+x_2^2+x_3^2
gap> form := QuadraticFormByPolynomial(poly, r);
< quadratic form >
gap> RadicalOfForm(form);
<vector space over GF(2^3), with 63 generators>
gap> r := PolynomialRing(GF(9),4);
GF(3^2)[x_1,x_2,x_3,x_4]
gap> poly := Z(3)^2*r.1^2+r.2^2+r.3*r.4;
x_1^2+x_2^2+x_3*x_4
gap> qform := QuadraticFormByPolynomial(poly,r);
< quadratic form >
gap> Display(qform);
Quadratic form
Gram Matrix:
 1 . . .
 . 1 . .
 . . . 1
 . . . .
Polynomial: x_1^2+x_2^2+x_3*x_4
```

4.3.3 HermitianFormByPolynomial

▷ `HermitianFormByPolynomial(`*poly, r[, n]*`)` (operation)

Returns: an hermitian form

The argument *poly* must be a polynomial in the polynomial ring *r* defined over a finite field of square order q^2 The (optional) last argument is the dimension for the underlying vector space of the resulting form, which by default is the number of indeterminates specified by *poly*. It is checked whether the polynomial is a homogeneous polynomial of degree $q+1$, and if not, an error message is returned. The output is a hermitian form in the category `IsHermitianForm`. (See 3.1 for more on hermitian forms).

Example

```
gap> r := PolynomialRing( GF(9), 4);
GF(3^2)[x_1,x_2,x_3,x_4]
gap> vars := IndeterminatesOfPolynomialRing( r );
[ x_1, x_2, x_3, x_4 ]
gap> poly := vars[1]*vars[2]^3+vars[1]^3*vars[2]+
>              vars[3]*vars[4]^3+vars[3]^3*vars[4];
x_1^3*x_2+x_1*x_2^3+x_3^3*x_4+x_3*x_4^3
gap> form := HermitianFormByPolynomial(poly,r);
< hermitian form >
gap> Display(form);
Hermitian form
Gram Matrix:
 . 1 . .
 1 . . .
 . . . 1
 . . 1 .
Polynomial: x_1^3*x_2+x_1*x_2^3+x_3^3*x_4+x_3*x_4^3
```

4.4 Switching between bilinear and quadratic forms

When the characteristic of the field is odd, a homogeneous quadratic polynomial determines a bilinear form, and a quadratic form. In some situations, when a quadratic form Q is given, it is useful to consider the bilinear form f such that $f(v,v) = Q(v)$, i.e., the bilinear form which is determined by exactly the same polynomial determining the quadratic form Q. Forms provides functionality to construct a bilinear form f from a given quadratic form Q such that $f(v,v) = Q(v)$. Conversely, we can extract a quadratic form from a given bilinear form.

4.4.1 QuadraticFormByBilinearForm

▷ `QuadraticFormByBilinearForm(`*form*`)` (operation)

Returns: a quadratic form

The argument *form* is an orthogonal bilinear form (and thus it belongs to `IsBilinearForm`), otherwise a "No method found" error is returned. The output is the quadratic form Q (an object in `IsQuadraticForm`), such that $Q(v) = form(v,v)$ for all vectors v in a vector space equipped with *form*. An error is returned when the characteristic of the field is even, or when *form* is not orthogonal.

Example

```
gap> mat := [ [ Z(3^2)^7, Z(3)^0, Z(3^2)^2, 0*Z(3), Z(3^2)^5 ],
>    [ Z(3)^0, Z(3^2)^7, Z(3^2)^6, Z(3^2)^5, Z(3^2)^2 ],
>    [ Z(3^2)^2, Z(3^2)^6, Z(3^2)^7, Z(3^2)^2, Z(3^2)^2 ],
>    [ 0*Z(3), Z(3^2)^5, Z(3^2)^2, Z(3^2)^6, Z(3^2)^7 ],
>    [ Z(3^2)^5, Z(3^2)^2, Z(3^2)^2, Z(3^2)^7, Z(3) ] ];
[ [ Z(3^2)^7, Z(3)^0, Z(3^2)^2, 0*Z(3), Z(3^2)^5 ],
  [ Z(3)^0, Z(3^2)^7, Z(3^2)^6, Z(3^2)^5, Z(3^2)^2 ],
  [ Z(3^2)^2, Z(3^2)^6, Z(3^2)^7, Z(3^2)^2, Z(3^2)^2 ],
  [ 0*Z(3), Z(3^2)^5, Z(3^2)^2, Z(3^2)^6, Z(3^2)^7 ],
  [ Z(3^2)^5, Z(3^2)^2, Z(3^2)^2, Z(3^2)^7, Z(3) ] ]
gap> form := BilinearFormByMatrix(mat,GF(9));
< bilinear form >
gap> Q := QuadraticFormByBilinearForm(form);
```

```
< quadratic form >
gap> Display(form);
Bilinear form
Gram Matrix:
z = Z(9)
 z^7   1 z^2   . z^5
   1 z^7 z^6 z^5 z^2
 z^2 z^6 z^7 z^2 z^2
   . z^5 z^2 z^6 z^7
 z^5 z^2 z^2 z^7   2
gap> Display(Q);
Quadratic form
Gram Matrix:
z = Z(9)
 z^7   2 z^6   . z^1
   . z^7 z^2 z^1 z^6
   .   . z^7 z^6 z^6
   .   .   . z^6 z^3
   .   .   .   .   2
gap> Set(List(GF(9)^5),x->[x,x]^form=x^Q);
[ true ]
gap> PolynomialOfForm(form);
Z(3^2)^7*x_1^2-x_1*x_2+Z(3^2)^6*x_1*x_3+Z(3^2)*x_1*x_5+Z(3^2)^7*x_2^2+Z(3^2)^2
*x_2*x_3+Z(3^2)*x_2*x_4+Z(3^2)^6*x_2*x_5+Z(3^2)^7*x_3^2+Z(3^2)^6*x_3*x_4+Z(3^2
)^6*x_3*x_5+Z(3^2)^6*x_4^2+Z(3^2)^3*x_4*x_5-x_5^2
gap> PolynomialOfForm(Q);
Z(3^2)^7*x_1^2-x_1*x_2+Z(3^2)^6*x_1*x_3+Z(3^2)*x_1*x_5+Z(3^2)^7*x_2^2+Z(3^2)^2
*x_2*x_3+Z(3^2)*x_2*x_4+Z(3^2)^6*x_2*x_5+Z(3^2)^7*x_3^2+Z(3^2)^6*x_3*x_4+Z(3^2
)^6*x_3*x_5+Z(3^2)^6*x_4^2+Z(3^2)^3*x_4*x_5-x_5^2
```

Note that the given bilinear form *form* is NOT the associated bilinear form of the constructed quadratic form Q, according to the definition in Section 3.2. We can construct the associated bilinear forms by using `AssociatedBilinearForm` (4.4.3). (See 3.2 for more on quadratic forms).

4.4.2 BilinearFormByQuadraticForm

▷ `BilinearFormByQuadraticForm(`*Q*`)` (operation)

Returns: a bilinear form

The argument Q must be a quadratic form (and thus it belongs to `IsQuadraticForm`). The output is the orthogonal bilinear form f (an object in `IsBilinearForm`), such that $f(v,v) = Q(v)$ for all vectors v in a vector space equipped with Q. An error is returned when the characteristic of the field is even.

Example

```
gap> r := PolynomialRing(GF(9),4);
GF(3^2)[x_1,x_2,x_3,x_4]
gap> poly := -r.1*r.2+Z(3^2)*r.3^2+r.4^2;
-x_1*x_2+Z(3^2)*x_3^2+x_4^2
gap> qform := QuadraticFormByPolynomial(poly,r);
< quadratic form >
gap> Display( qform );
Quadratic form
```

```
Gram Matrix:
z = Z(9)
   .   2   .   .
   .   .   .   .
   .   . z^1   .
   .   .   .   1
Polynomial: -x_1*x_2+Z(3^2)*x_3^2+x_4^2
gap> form := BilinearFormByQuadraticForm( qform );
< bilinear form >
gap> Display(form);
Bilinear form
Gram Matrix:
z = Z(9)
   .   1   .   .
   1   .   .   .
   .   . z^1   .
   .   .   .   1
gap> Set(GF(9)^4, x -> [x,x]^form = x^qform);
[ true ]
```

Note that the constructed bilinear form f is NOT the associated bilinear form of the given quadratic form Q, according to the definition in Section 3.2. We can construct the associated bilinear forms by using `AssociatedBilinearForm` (4.4.3). (See 3.2 for more on quadratic forms).

4.4.3 AssociatedBilinearForm

▷ `AssociatedBilinearForm(`*Q*`)` (operation)

Returns: a bilinear form

The argument Q must be a quadratic form (and thus it belongs to `IsQuadraticForm`). The output is the associated bilinear form f (an object in `IsBilinearForm`), as defined in Section 3.2, i.e. the bilinear form f such that $f(v,w) = Q(v+w) - Q(v) - Q(w)$ for all vectors v,w in a vector space equipped with Q. (See 3.2 for more on quadratic forms).

Example

```
gap> r:= PolynomialRing(GF(121),6);
GF(11^2)[x_1,x_2,x_3,x_4,x_5,x_6]
gap> poly := r.1*r.5-r.2*r.6+r.3*r.4;
x_1*x_5-x_2*x_6+x_3*x_4
gap> form := QuadraticFormByPolynomial(poly,r);
< quadratic form >
gap> aform := AssociatedBilinearForm(form);
< bilinear form >
gap> Display(aform);
Bilinear form
Gram Matrix:
  .  .  .  .  1  .
  .  .  .  .  . 10
  .  .  .  1  .  .
  .  .  1  .  .  .
  1  .  .  .  .  .
  . 10  .  .  .  .
```

4.5 Evaluating forms

4.5.1 EvaluateForm

▷ `EvaluateForm(f, u[, v])` (operation)

Returns: a finite field element

The argument `f` is either a sesquilinear or quadratic form defined over a finite field GF(q). The other argument is a pair of vectors or matrices, or a single vector or matrix. In case that `u` (and `v` when using three arguments) is a matrix, its rows represent a basis for the subspace (or subspaces) where f is evaluated in. This operation evaluates the form on the given vector or pair of vectors and returns an element in $GF(q)$. There is also an overloading of the operation `\^` where `(u,v)^f` represents $f(u,v)$ in the case that `f` is sesquilinear, and `u^f` stands for $f(u)$ in the quadratic case. So for convenience, the user may use this compressed version of this operation, which we show in the following example:

Example

```
gap> mat := [[Z(8),0,0,0],[0,0,Z(8)^4,0],[0,0,0,1],[0,0,0,0]]*Z(8)^0;;
gap> form := QuadraticFormByMatrix(mat,GF(8));
< quadratic form >
gap> u := [ Z(2^3)^4, Z(2^3)^4, Z(2)^0, Z(2^3)^3 ];
[ Z(2^3)^4, Z(2^3)^4, Z(2)^0, Z(2^3)^3 ]
gap> EvaluateForm( form, u );
Z(2^3)^6
gap> u^form;
Z(2^3)^6
gap> gram := [[0,0,0,0,0,2],[0,0,0,0,2,0],[0,0,0,1,0,0],
>               [0,0,1,0,0,0],[0,2,0,0,0,0],[2,0,0,0,0,0]]*Z(3)^0;;
gap> form := BilinearFormByMatrix(gram,GF(3));
< bilinear form >
gap> u := [ [ Z(3)^0, 0*Z(3), 0*Z(3), Z(3)^0, 0*Z(3), Z(3)^0 ],
>   [ 0*Z(3), 0*Z(3), Z(3)^0, Z(3)^0, Z(3), 0*Z(3) ] ];;
gap> v := [ [ Z(3)^0, 0*Z(3), Z(3)^0, Z(3), 0*Z(3), Z(3) ],
>   [ 0*Z(3), Z(3)^0, 0*Z(3), Z(3), Z(3), Z(3) ] ];;
gap> EvaluateForm( form, u, v);
[ [ Z(3)^0, Z(3)^0 ], [ 0*Z(3), 0*Z(3) ] ]
gap> [u,v]^form;
[ [ Z(3)^0, Z(3)^0 ], [ 0*Z(3), 0*Z(3) ] ]
```

4.6 Orthogonality, totally isotropic subspaces, and totally singular subspaces

4.6.1 OrthogonalSubspaceMat (for a vector)

▷ `OrthogonalSubspaceMat(form, v)` (operation)

▷ `OrthogonalSubspaceMat(form, mat)` (operation)

Returns: a base of the subspace orthogonal to the given vector or subspace with relation to the given form

The argument `form` is a sesquilinear or quadratic form. For a given vector `v`, this operation returns a base of the subspace orthogonal to `v` with relation to the sesquilinear `form` or with relation to the associated bilinear form of the quadratic form `form`. For a given matrix `mat`, this operation returns

a base of the subspace orthogonal to the subspace spanned by the rows of *mat* with relation to the sesquilinear *form* or with relation to the associated bilinear form of the quadratic form *form*

Example

```
gap> mat := [[0,0,0,-2],[0,0,-3,0],[0,3,0,0],[2,0,0,0]]*Z(7)^0;
[ [ 0*Z(7), 0*Z(7), 0*Z(7), Z(7)^5 ], [ 0*Z(7), 0*Z(7), Z(7)^4, 0*Z(7) ],
  [ 0*Z(7), Z(7), 0*Z(7), 0*Z(7) ], [ Z(7)^2, 0*Z(7), 0*Z(7), 0*Z(7) ] ]
gap> form := BilinearFormByMatrix(mat);
< bilinear form >
gap> v := [0*Z(7),Z(7)^0,Z(7)^3,Z(7)^5];
[ 0*Z(7), Z(7)^0, Z(7)^3, Z(7)^5 ]
gap> vperp := OrthogonalSubspaceMat(form,v);
[ [ Z(7)^0, Z(7)^0, 0*Z(7), 0*Z(7) ], [ Z(7)^0, 0*Z(7), Z(7)^0, 0*Z(7) ],
  [ 0*Z(7), 0*Z(7), 0*Z(7), Z(7)^0 ] ]
gap> List(vperp,x->[x,v]^form);
[ 0*Z(7), 0*Z(7), 0*Z(7) ]
gap> sub := [[1,1,0,0],[0,0,1,2]]*Z(7)^0;
[ [ Z(7)^0, Z(7)^0, 0*Z(7), 0*Z(7) ], [ 0*Z(7), 0*Z(7), Z(7)^0, Z(7)^2 ] ]
gap> subperp := OrthogonalSubspaceMat(form,sub);
[ [ Z(7)^0, Z(7)^0, 0*Z(7), 0*Z(7) ], [ 0*Z(7), 0*Z(7), Z(7)^4, Z(7)^0 ] ]
gap> List(subperp,x->List(sub,y->[x,y]^form));
[ [ 0*Z(7), 0*Z(7) ], [ 0*Z(7), 0*Z(7) ] ]
gap> mat := [[1,0,0],[0,0,1],[0,0,0]]*Z(2)^0;
[ [ Z(2)^0, 0*Z(2), 0*Z(2) ], [ 0*Z(2), 0*Z(2), Z(2)^0 ],
  [ 0*Z(2), 0*Z(2), 0*Z(2) ] ]
gap> form := QuadraticFormByMatrix(mat);
< quadratic form >
gap> v := [Z(2)^0,Z(2)^0,0*Z(2)];
[ Z(2)^0, Z(2)^0, 0*Z(2) ]
gap> vperp := OrthogonalSubspaceMat(form,v);
[ <an immutable GF2 vector of length 3>, <an immutable GF2 vector of length
    3> ]
gap> bil_form := AssociatedBilinearForm(form);
< bilinear form >
gap> List(vperp,x->[x,v]^bil_form);
[ 0*Z(2), 0*Z(2) ]
gap> sub := [[1,0,1],[1,0,0]]*Z(2)^0;
[ [ Z(2)^0, 0*Z(2), Z(2)^0 ], [ Z(2)^0, 0*Z(2), 0*Z(2) ] ]
gap> subperp := OrthogonalSubspaceMat(form,sub);
[ <an immutable GF2 vector of length 3>, <an immutable GF2 vector of length
    3> ]
gap> List(subperp,x->List(sub,y->[x,y]^bil_form));
[ [ 0*Z(2), 0*Z(2) ], [ 0*Z(2), 0*Z(2) ] ]
```

4.6.2 IsIsotropicVector

▷ `IsIsotropicVector(`*form*`, `*v*`)` (operation)

Returns: true or false

The operation return *true* if and only if *v* is isotropic with relation to the sesquilinear or quadratic form *form*.

Example

```
gap> mat := [[1,0,0,0],[0,-1,0,0],[0,0,0,1],[0,0,1,0]]*Z(41)^0;
```

```
[ [ Z(41)^0, 0*Z(41), 0*Z(41), 0*Z(41) ],
  [ 0*Z(41), Z(41)^20, 0*Z(41), 0*Z(41) ],
  [ 0*Z(41), 0*Z(41), 0*Z(41), Z(41)^0 ],
  [ 0*Z(41), 0*Z(41), Z(41)^0, 0*Z(41) ] ]
gap> form := BilinearFormByMatrix(mat);
< bilinear form >
gap> v := [1,1,0,0]*Z(41)^0;
[ Z(41)^0, Z(41)^0, 0*Z(41), 0*Z(41) ]
gap> IsIsotropicVector(form,v);
true
gap> mat := [[1,0,0,0,0],[0,0,0,0,1],[0,0,0,0,0],[0,0,1,0,0],[0,0,0,0,0]]*Z(8)^0;
[ [ Z(2)^0, 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2) ],
  [ 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2), Z(2)^0 ],
  [ 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2) ],
  [ 0*Z(2), 0*Z(2), Z(2)^0, 0*Z(2), 0*Z(2) ],
  [ 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2) ] ]
gap> form := QuadraticFormByMatrix(mat);
< quadratic form >
gap> v1 := [1,0,0,0,0]*Z(8)^0;
[ Z(2)^0, 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2) ]
gap> v2 := [0,1,0,0,0]*Z(8)^0;
[ 0*Z(2), Z(2)^0, 0*Z(2), 0*Z(2), 0*Z(2) ]
gap> IsIsotropicVector(form,v1);
true
gap> IsIsotropicVector(form,v2);
true
```

4.6.3 IsSingularVector

▷ `IsSingularVector(`*form*`, `*v*`)` (operation)

Returns: true or false

The operation return *true* if and only if *v* is singular with relation to the quadratic form *form*. Note that only when the characteristic of the field is odd, the singular vectors with relation to a quadratic form are the isotropic vectors with relation to its associated form.

Example

```
gap> mat := [[1,0,0,0,0],[0,0,0,0,1],[0,0,0,0,0],[0,0,1,0,0],[0,0,0,0,0]]*Z(8)^0;
[ [ Z(2)^0, 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2) ],
  [ 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2), Z(2)^0 ],
  [ 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2) ],
  [ 0*Z(2), 0*Z(2), Z(2)^0, 0*Z(2), 0*Z(2) ],
  [ 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2) ] ]
gap> form := QuadraticFormByMatrix(mat);
< quadratic form >
gap> v1 := [1,0,0,0,0]*Z(8)^0;
[ Z(2)^0, 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2) ]
gap> v2 := [0,1,0,0,0]*Z(8)^0;
[ 0*Z(2), Z(2)^0, 0*Z(2), 0*Z(2), 0*Z(2) ]
gap> IsSingularVector(form,v1);
false
gap> IsSingularVector(form,v2);
```

```
true
gap> IsIsotropicVector(form,v1);
true
gap> IsIsotropicVector(form,v2);
true
```

4.6.4 IsTotallyIsotropicSubspace

▷ IsTotallyIsotropicSubspace(*form*, *sub*) (operation)

Returns: true or false

The operation return *true* if and only if the subspace spanned by the vectors in the list *sub* is totally isotropic with relation to the sesquilinear or quadratic form *form*. Note that when *form* is a quadratic form, it is checked whether *sub* generates a subspace that is totally isotropic with relation to the associated bilinear form of *form*.

Example

```
gap> mat := [[1,0,0,0],[0,-1,0,0],[0,0,0,1],[0,0,1,0]]*Z(7)^0;
[ [ Z(7)^0, 0*Z(7), 0*Z(7), 0*Z(7) ], [ 0*Z(7), Z(7)^3, 0*Z(7), 0*Z(7) ],
  [ 0*Z(7), 0*Z(7), 0*Z(7), Z(7)^0 ], [ 0*Z(7), 0*Z(7), Z(7)^0, 0*Z(7) ] ]
gap> form := BilinearFormByMatrix(mat);
< bilinear form >
gap> sub:= [[Z(7)^0,0*Z(7),Z(7)^0,Z(7)],[0*Z(7),Z(7)^0,Z(7)^0,Z(7)^4]];
[ [ Z(7)^0, 0*Z(7), Z(7)^0, Z(7) ], [ 0*Z(7), Z(7)^0, Z(7)^0, Z(7)^4 ] ]
gap> IsTotallyIsotropicSubspace(form,sub);
true
gap> mat := IdentityMat(6,GF(2));
[ <a GF2 vector of length 6>, <a GF2 vector of length 6>,
  <a GF2 vector of length 6>, <a GF2 vector of length 6>,
  <a GF2 vector of length 6>, <a GF2 vector of length 6> ]
gap> form := HermitianFormByMatrix(mat,GF(4));
< hermitian form >
gap> sub := [[Z(2)^0,0*Z(2),0*Z(2),Z(2)^0,Z(2)^0,Z(2)^0],
>   [0*Z(2),Z(2)^0,0*Z(2),Z(2^2)^2,Z(2^2),Z(2)^0],
>   [0*Z(2),0*Z(2),Z(2)^0,Z(2)^0,Z(2^2),Z(2^2)^2]];
[ [ Z(2)^0, 0*Z(2), 0*Z(2), Z(2)^0, Z(2)^0, Z(2)^0 ],
  [ 0*Z(2), Z(2)^0, 0*Z(2), Z(2^2)^2, Z(2^2), Z(2)^0 ],
  [ 0*Z(2), 0*Z(2), Z(2)^0, Z(2)^0, Z(2^2), Z(2^2)^2 ] ]
gap> IsTotallyIsotropicSubspace(form,sub);
true
```

4.6.5 IsTotallySingularSubspace

▷ IsTotallySingularSubspace(*form*, *sub*) (operation)

Returns: true or false

The operation return *true* if and only if the subspace spanned by the vectors in the list *sub* is totally singular with relation to quadratic form *form*. Note that only when the characteristic of the field is odd, the totally singular subspaces of given dimension n with relation to a quadratic form are exactly the totally isotropic subspaces of dimension n with relation to its associated form.

Example

```
gap> mat := [[1,0,0,0,0],[0,0,0,0,1],[0,0,0,0,0],[0,0,1,0,0],[0,0,0,0,0]]*Z(8)^0;
[ [ Z(2)^0, 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2) ],
  [ 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2), Z(2)^0 ],
  [ 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2) ],
  [ 0*Z(2), 0*Z(2), Z(2)^0, 0*Z(2), 0*Z(2) ],
  [ 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2) ] ]
gap> form := QuadraticFormByMatrix(mat);
< quadratic form >
gap> sub := [[Z(2)^0,0*Z(2),Z(2^3)^6,Z(2^3),Z(2^3)^3],
>        [0*Z(2),Z(2)^0,Z(2^3)^6,Z(2^3)^2,Z(2^3)]];
[ [ Z(2)^0, 0*Z(2), Z(2^3)^6, Z(2^3), Z(2^3)^3 ],
  [ 0*Z(2), Z(2)^0, Z(2^3)^6, Z(2^3)^2, Z(2^3) ] ]
gap> IsTotallySingularSubspace(form,sub);
true
```

4.7 Attributes and properties of forms

4.7.1 IsReflexiveForm

▷ `IsReflexiveForm(f)` (property)

Returns: true or false.

A sesquilinear form f on a vector space V is *reflexive* if $f(v,w) = 0 \Rightarrow f(w,v) = 0$ for all $v,w \in V$. The argument f must be a sesquilinear form (and thus it belongs to `IsSesquilinearForm`). A sesquilinear form f is *reflexive* if whenever we have $f(u,v) = 0$, for two vectors u,v in the associated vector space, then we also have $f(v,u) = 0$. This attribute simply returns `true` or `false` according to whether `f` is reflexive or not, and is stored as a property of `f`. It is not possible in this version of Forms to construct non-reflexive forms. (See 3.1 for more on reflexive sesquilinear forms).

4.7.2 IsAlternatingForm

▷ `IsAlternatingForm(f)` (property)

Returns: true or false.

A sesquilinear form f on a vector space V is *alternating* if $f(v,v) = 0$ for all $v \in V$. The argument f must be a sesquilinear form (and thus it belongs to `IsSesquilinearForm`). A bilinear form f is *alternating* if $f(v,v) = 0$ for all v. This method simply returns `true` or `false` according to whether `f` is alternating or not, and is stored as a property of `f`. (See 3.1 for more on alternating sesquilinear forms).

4.7.3 IsSymmetricForm

▷ `IsSymmetricForm(f)` (property)

Returns: true or false.

A sesquilinear form f on a vector space V is *symmetric* if $f(v,w) = f(w,v)$ for all $v,w \in V$. The argument f must be a sesquilinear form (and thus it belongs to `IsSesquilinearForm`). A bilinear form f is *symmetric* if $f(u,v) = f(v,u)$ for all pairs of vectors u and v. This attribute simply returns

`true` or `false` according to whether `f` is symmetric or not, and is stored as a property of `f`. (See 3.1 for more on symmetric sesquilinear forms).

4.7.4 IsOrthogonalForm

▷ `IsOrthogonalForm(f)` (property)

Returns: true or false.

The argument f must be a sesquilinear form (and thus it belongs to `IsSesquilinearForm`). A bilinear form f is called *orthogonal* if the characteristic of the underlying field is odd, and f is a symmetric form. (See 3.1 for more on bilinear forms). This operation simply returns `true` or `false` according to whether `f` is an orthogonal bilinear form or not, and is stored as a property of `f`.

4.7.5 IsPseudoForm

▷ `IsPseudoForm(f)` (property)

Returns: true or false.

When the characteristic of the field is odd, we call a form f *orthogonal* if and only f is symmetric, and when the characteristic of the field is even, we call a form f *pseudo* if and only if f is symmetric but not alternating. The argument f must be a sesquilinear form (and thus it belongs to `IsSesquilinearForm`). (See 3.1 for more on pseudo forms). This method simply returns `true` or `false` according to whether `f` is a pseudo form or not, and is stored as a property of `f`.

4.7.6 IsSymplecticForm

▷ `IsSymplecticForm(f)` (property)

Returns: true or false.

We call a bilinear form f *symplectic* if and only if f is alternating. The argument f must be a sesquilinear form (and thus it belongs to `IsSesquilinearForm`). (See 3.1 for more on symplectic forms). This method simply returns `true` or `false` according to whether `f` is symplectic or not, and is stored as a property of `f`.

4.7.7 IsDegenerateForm

▷ `IsDegenerateForm(f)` (property)

Returns: true or false.

The argument f must be a form (and thus it belongs to `IsForm`). A sesquilinear form f is *degenerate* if its radical is non-trivial. A quadratic form is degenerate if and only if the radical of the associated bilinear form is non-trivial. Note that degeneracy for quadratic forms is too restrictive if the characteristic is even. See also `IsSingularForm` (4.7.8). This attribute simply returns `true` or `false` according to whether `f` is degenerate or not, and is stored as a property of `f`.

4.7.8 IsSingularForm

▷ `IsSingularForm(f)` (property)

Returns: true or false.

The argument f must be a quadratic form (and thus it belongs to `IsQuadraticForm`). A quadratic form f is *singular* if its radical is non-trivial. When the characteristic of the field is odd, a quadratic form is singular if and only if it is degenerate. This is not the case when the characteristic of the field

is even. This method simply returns `true` or `false` according to whether `f` is singular or not, and is stored as a property of `f`.

4.7.9 BaseField

▷ `BaseField(f)` (attribute)

Returns: the underlying field of `f`.

The argument f must be a form (and thus it belongs to `IsForm`). The method returns the field which is stored as the *defining field* of f. We sometimes stipulate in Forms that a form have a defining field, for mathematical reasons. Clearly, to define a hermitian form one needs to specify the field of scalars for the vector space that you wish your hermitian form to act on. The default, if the user has not specified a field on creation of a form, is the smallest field containing the entries or coefficients of the input (a matrix or polynomial). Having a particular defining field for a form can be very useful, for example, when one wants to find a change of basis from one form to another (isometric) form. In this case, one needs to know in which $\mathrm{GL}(d,q)$ the base-transition matrix should be taken.

4.7.10 GramMatrix

▷ `GramMatrix(f)` (attribute)

Returns: the Gram matrix of `f`.

The argument f must be a form (and thus it belongs to `IsForm`). This method returns the Gram matrix of f (see 3.1 and 3.2).

4.7.11 RadicalOfForm

▷ `RadicalOfForm(f)` (attribute)

Returns: The radical of the form `f`

The argument f must be a form (and thus it belongs to `IsForm`) on some vector space V. The radical of a sesquilinear form f is the subspace consisting of vectors which are orthogonal to every vector, i.e.,

$$\mathrm{Rad}(f) = \{v \in V | f(v,w) = 0, \forall w \in V\}.$$

The *radical* of the quadratic form Q, is the intersection of the set of all singular vectors with relation to Q and the radical of the associated bilinear form f (see `AssociatedBilinearForm` (4.4.3)), i.e.

$$\mathrm{Rad}(Q) = \{v \in V | Q(v) = 0 \text{ and } v \in \mathrm{Rad}(f)\}.$$

Example

```
gap> r := PolynomialRing( GF(8), 3 );
GF(2^3)[x_1,x_2,x_3]
gap> poly := r.1^2 + r.2 * r.3;
x_1^2+x_2*x_3
gap> form := QuadraticFormByPolynomial( poly, r );
< quadratic form >
gap> r := RadicalOfForm( form );
<vector space over GF(2^3), with 0 generators>
gap> Dimension(r);
0
```

4.7.12 PolynomialOfForm

▷ `PolynomialOfForm(f)` (attribute)

Returns: the polynomial associated with `f`.

The argument `f` must be a form (and thus it belongs to `IsForm`). All forms, except for bilinear forms in even characteristic, have an associated polynomial defining a quadratic or hermitian form (see 3.1 and 3.2). This method returns the polynomial associated with `f`, and if not already bound, stores it as a property of `f`.

Example

```
gap> mat := [ [ Z(8) , 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2) ],
>  [ 0*Z(2), Z(2)^0, Z(2^3)^5, 0*Z(2), 0*Z(2) ],
>  [ 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2) ],
>  [ 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2), Z(2)^0 ],
>  [ 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2) ] ];;
gap> form := QuadraticFormByMatrix(mat,GF(8));
< quadratic form >
gap> PolynomialOfForm(form);
Z(2^3)*x_1^2+x_2^2+Z(2^3)^5*x_2*x_3+x_4*x_5
```

4.7.13 DiscriminantOfForm

▷ `DiscriminantOfForm(f)` (attribute)

Returns: a string

The argument f must be a form (and thus it belongs to `IsForm`). Given a quadratic or bilinear form `f` of even dimension, this operation returns a string: "square" or "nonsquare". More specifically, let f be a from over GF(q), and let M be the Gram matrix of f. Define the *discriminant* of Q (n.b., *quasideterminant* in [CCN+85]) as 'square' if $det(M)$ is a square of GF(q), and 'non-square' otherwise. The discriminant is an invariant of nondegenerate orthogonal spaces over finite fields of odd order, up to isometry. Thus, discriminants can be used to delineate the isometry type of an orthogonal form in even (algebraic) dimension. The discriminant of a hermitian form is not defined, and applying this operation on a hermitian form, will result in an error message.

Example

```
gap> gram := InvariantQuadraticForm(GO(-1,4,5))!.matrix;;
gap> qform := QuadraticFormByMatrix(gram, GF(5));
< quadratic form >
gap> DiscriminantOfForm( qform );
"nonsquare"
```

4.8 Recognition of forms preserved by a classical group

In this section, we describe a function that was initially developed by Frank Celler (and which has now been adapted to Forms) for the recognition of quadratic and sesquilinear forms left invariant by a matrix group. More importantly, we should stress that this routine differs to that already offered by the MeatAxe in that it finds sesquilinear forms preserved up to SCALARS. Eventually, the procedure used for finding preserved sesquilinear forms does use the MeatAxe but in some cases it can rule out the existence of preserved forms without calling the MeatAxe. For more information on the algorithm, please see [CLGN+08].

4.8.1 PreservedForms

▷ `PreservedForms(`*group*`)` (operation)

Returns: a list of forms

The argument *group* is a matrix group. The function uses random methods to find all of the bilinear, unitary or quadratic forms preserved by *group* (the trivial form is also a possibility) up to a scalar. Since the procedure relies on a pseudo-random generator, the user may need to execute the operation more than once to find all invariant sesquilinear forms. Note that when possible, a quadratic form will be given.

Example

```
gap> group := GO(-1,4,4);
GO(-1,4,4)
gap> pres_forms := PreservedForms(group);
[ < quadratic form > ]
gap> group := GO(1,6,9);
GO(+1,6,9)
gap> pres_forms := PreservedForms(group);
[ < quadratic form > ]
gap> group := SU(4,16);
SU(4,16)
gap> pres_forms := PreservedForms(group);
[ < hermitian form > ]
```

The next example demonstrates the impact of randomized methods on the number of preserved forms returned. For the particular matrix group, two preserved forms are returned after four calls of the operation.

Example

```
gap> gens := [ [ [ Z(5)^0, 0*Z(5), 0*Z(5), 0*Z(5) ],
>          [ 0*Z(5), 0*Z(5), Z(5)^3, Z(5^2)^21 ],
>          [ 0*Z(5), Z(5), Z(5), Z(5^2)^3 ],
>              [ 0*Z(5), Z(5^2)^21, Z(5^2)^15, Z(5)^2 ] ],
>    [ [ Z(5)^3, Z(5^2)^7, Z(5^2)^16, Z(5^2)^15 ],
>                [ 0*Z(5), Z(5)^0, Z(5^2)^4, Z(5)^3 ],
>                [ Z(5^2)^22, Z(5^2)^10, Z(5^2)^21, Z(5)^2 ],
>                [ Z(5^2)^7, Z(5^2)^23, Z(5^2)^9, Z(5^2)^11 ] ],
>    [ [ 0*Z(5), Z(5^2), 0*Z(5), 0*Z(5) ],
>          [ Z(5^2)^5, 0*Z(5), 0*Z(5), 0*Z(5) ],
>                [ 0*Z(5), 0*Z(5), Z(5)^0, Z(5^2)^4 ],
>                [ 0*Z(5), 0*Z(5), Z(5^2)^20, Z(5)^2 ] ] ];
[ [ [ Z(5)^0, 0*Z(5), 0*Z(5), 0*Z(5) ], [ 0*Z(5), 0*Z(5), Z(5)^3, Z(5^2)^21 ],
      [ 0*Z(5), Z(5), Z(5), Z(5^2)^3 ],
      [ 0*Z(5), Z(5^2)^21, Z(5^2)^15, Z(5)^2 ] ],
  [ [ Z(5)^3, Z(5^2)^7, Z(5^2)^16, Z(5^2)^15 ],
      [ 0*Z(5), Z(5)^0, Z(5^2)^4, Z(5)^3 ],
      [ Z(5^2)^22, Z(5^2)^10, Z(5^2)^21, Z(5)^2 ],
      [ Z(5^2)^7, Z(5^2)^23, Z(5^2)^9, Z(5^2)^11 ] ],
  [ [ 0*Z(5), Z(5^2), 0*Z(5), 0*Z(5) ], [ Z(5^2)^5, 0*Z(5), 0*Z(5), 0*Z(5) ],
      [ 0*Z(5), 0*Z(5), Z(5)^0, Z(5^2)^4 ],
      [ 0*Z(5), 0*Z(5), Z(5^2)^20, Z(5)^2 ] ] ]
gap> group := Group(gens);
<matrix group with 3 generators>
```

```
gap> PreservedForms(group);
[ < quadratic form > ]
gap> PreservedForms(group);
[ < quadratic form > ]
gap> PreservedForms(group);
[ < quadratic form > ]
gap> PreservedForms(group);
[ < quadratic form >, < hermitian form > ]
```

4.8.2 PreservedSesquilinearForms

▷ PreservedSesquilinearForms(*group*) (operation)

Returns: a list of forms

The argument *group* is a matrix group. The function uses random methods to find all of the bilinear or unitary forms preserved by *group* (the trivial form is also a possibility) up to a scalar. Since the procedure relies on a pseudo-random generator, the user may need to execute the operation more than once to find all invariant sesquilinear forms.

Example

```
gap> g := SU(4,3);
SU(4,3)
gap> forms := PreservedSesquilinearForms(g);
[ < hermitian form > ]
gap> Display( forms[1] );
Hermitian form
Gram Matrix:
 . . . 2
 . . 2 .
 . 2 . .
 2 . . .
```

Here is another example which shows that this procedure is suitable in some cases where using the MeatAxe is not applicable. Here, our matrix group is the group of similarities preserving a (hyperbolic) bilinear form on a 6-dimensional vector space over GF(3) .

Example

```
gap> a := [ [ -1, 0, 0, -1, 0, 1 ], [ 0, -1, -1, 0, 0, 1 ],
>        [ -1, 0, 0, 1, 0, 0 ],  [ 0, -1, 1, 0, 0, -1 ],
>        [ 0, 0, 0, 0, 0, -1 ], [ 0, -1, -1, 1, 1, 1 ] ] * One(GF(3));;
gap> b := [ [ 1, -1, 1, -1, 1, -1 ], [ 1, 1, -1, 1, 1, 0 ],
>        [ -1, 0, 1, 0, 0, 0 ], [ 0, -1, 0, 0, 0, 1 ],
>        [ 1, 1, 1, 1, 1, 1 ], [ -1, 1, 1, 1, -1, 0 ] ] * One(GF(3));;
gap> g := Group( a, b );
<matrix group with 2 generators>
gap> forms := PreservedSesquilinearForms( g );
[ < bilinear form > ]
gap> Display( forms[1] );
Bilinear form
Gram Matrix:
 . 1 . . . .
 1 . . . . .
 . . . 1 . .
```

```
 . . 1 . . .
 . . . . . 1
 . . . . 1 .
gap> m := GModuleByMats( [a,b], GF(3) );;
gap> usemeataxe := MTX.InvariantBilinearForm(m);
fail
```

4.8.3 PreservedQuadraticForms

▷ PreservedQuadraticForms(*group*) (operation)

Returns: a list of forms

The argument *group* is a matrix group. The function uses random methods to find all of the quadratic forms preserved by *group* up to a scalar. Since the procedure relies on a pseudo-random generator, the user may need to execute the operation more than once to find all invariant sesquilinear forms.

Example

```
gap> group := GO(-1,4,8);
GO(-1,4,8)
gap> pres_forms := PreservedQuadraticForms(group);
[ < quadratic form > ]
gap> group := GO(1,6,9);
GO(+1,6,9)
gap> pres_forms := PreservedQuadraticForms(group);
[ < quadratic form > ]
```

4.9 The trivial form and some of its properties

It can be useful to work with trivial a quadratic or sesquilinear form, i.e. a form mapping all vectors, couples of vectors respectively, to the zero element of their basefield. As mentioned in Section 4.1, Forms allows the construction of an object in the Category IsTrivialForm.

Example

```
gap> mat := [[0,0,0],[0,0,0],[0,0,0]]*Z(7)^0;
[ [ 0*Z(7), 0*Z(7), 0*Z(7) ], [ 0*Z(7), 0*Z(7), 0*Z(7) ],
  [ 0*Z(7), 0*Z(7), 0*Z(7) ] ]
gap> form1 := BilinearFormByMatrix(mat,GF(7));
< trivial form >
gap> form2 := QuadraticFormByMatrix(mat,GF(7));
< trivial form >
gap> form1 = form2;
true
gap> IsQuadraticForm(form1);
false
gap> IsSesquilinearForm(form1);
false
gap> mat := [[0,0],[0,0]]*Z(4)^0;
[ [ 0*Z(2), 0*Z(2) ], [ 0*Z(2), 0*Z(2) ] ]
gap> form3 := BilinearFormByMatrix(mat,GF(4));
< trivial form >
gap> form3 = form1;
```

```
false
```

As we have seen by the above example, there is only one trivial form for a given vector space over a finite field, and such a trivial form can result from the construction of a quadratic form or a sesquilinear form, but the trivial form itself is none of these, although it can behave as a sesquilinear or a quadratic form, depending on its arguments.

Example

```
gap> mat := [[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0]]*Z(3)^0;
[ [ 0*Z(3), 0*Z(3), 0*Z(3), 0*Z(3) ], [ 0*Z(3), 0*Z(3), 0*Z(3), 0*Z(3) ],
  [ 0*Z(3), 0*Z(3), 0*Z(3), 0*Z(3) ], [ 0*Z(3), 0*Z(3), 0*Z(3), 0*Z(3) ] ]
gap> form := BilinearFormByMatrix(mat,GF(3));
< trivial form >
gap> v := Random(GF(3)^4);
[ Z(3), Z(3), 0*Z(3), Z(3) ]
gap> [v,v]^form;
0*Z(3)
gap> v^form;
0*Z(3)
```

The attributes and properties described in Section 4.7 are all applicable to trivial forms.

Example

```
gap> mat := [[0,0,0],[0,0,0],[0,0,0]]*Z(11)^0;
[ [ 0*Z(11), 0*Z(11), 0*Z(11) ], [ 0*Z(11), 0*Z(11), 0*Z(11) ],
  [ 0*Z(11), 0*Z(11), 0*Z(11) ] ]
gap> form := QuadraticFormByMatrix(mat,GF(121));
< trivial form >
gap> IsReflexiveForm(form);
true
gap> IsAlternatingForm(form);
true
gap> IsSymmetricForm(form);
true
gap> IsOrthogonalForm(form);
false
gap> IsPseudoForm(form);
false
gap> IsSymplecticForm(form);
true
gap> IsDegenerateForm(form);
true
gap> IsSingularForm(form);
true
gap> BaseField(form);
GF(11^2)
gap> GramMatrix(form);
[ [ 0*Z(11), 0*Z(11), 0*Z(11) ], [ 0*Z(11), 0*Z(11), 0*Z(11) ],
  [ 0*Z(11), 0*Z(11), 0*Z(11) ] ]
gap> RadicalOfForm(form);
<vector space over GF(11^2), with 3 generators>
```

Chapter 5

Morphisms of forms

In this chapter we give a very brief overview on morphisms of sesquilinear and quadratic forms. The reader can find more in the texts: Cameron [Cam00], Taylor [Tay92], Aschbacher [Asc00], or Kleidman and Liebeck [KL90].

In this chapter we consider an n-dimensional vector space V over a finite field. Suppose that f is a sesquilinear form or a quadratic form on V, then we call the pair (V, f) a *formed vector space*.

5.1 Morphisms of sesquilinear forms

Consider two formed vector spaces (V, f) and (W, g) over the same field F, where both f and g are sesquilinear forms. Suppose that ϕ is a linear map from V to W. The map ϕ is an *isometry* from the formed space (V, f) to the formed space (W, g) if for all $v, w \in V$ we have

$$f(v, w) = f'(\phi(v), \phi(w)).$$

The map ϕ is a *similarity* from the formed space (V, f) to the formed space (W, g) if for all $v, w \in V$ we have

$$f(v, w) = \lambda f'(\phi(v), \phi(w)).$$

for some non-zero $\lambda \in F$. Finally, the map ϕ is a *semi-similarity* from the formed space (V, f) to the formed space (W, g) if for all v, w in V we have

$$f(v, w) = \lambda f'(\phi(v), \phi(w))^{\alpha}$$

for some non-zero $\lambda \in F$ and a field automorphism α of F.

One of the objectives of studying maps between formed vector spaces is the classification of sesquilinear forms on a vector space V, where it is sufficient to classify non-degenerate forms. The following results are well known.

It can be proved that (see for example Section 6.3 of [Cam00]):

- all non-degenerate alternating forms of a given vector space over a given finite field are similar,
- all non-degenerate hermitian forms of a given vector space over a given finite field are similar, and,
- the non-degenerate symmetric bilinear forms on a vector space over a field with odd characteristic come in three flavours, two of which occur when the dimension of the vector space is even, one of which occurs when the dimension of the vector space is odd.

In principle, within each similarity class, different isometry classes can occur, but we will see that in most cases, each similarity class contains exactly one isometry class.

Given a sesquilinear form f over a vector space V, Forms provides functionality to compute the linear map ϕ from V to itself (or, equivalently, a matrix describing a change of basis), such that f is mapped to its canonical representative in its isometry class. In the next sections, we describe the representative(s) of the similarity class(es) used in Forms, and, when necessary, the different isometry classes, for each of the three reflexive sesquilinear forms. The easiest cases are the hermitian and alternating cases.

5.1.1 Hermitian forms

We suppose that f is a non-degenerate hermitian form on a vector space V over the finite field F, with involutory field automorphism α. It can be proved (see [KL90]) that any vector space equipped with a non-degenerate hermitian form f contains an orthogonal basis such that $f(e_i, e_i) = 1$ for each basisvector e_i . Hence (V, f) is isometric with (V, f') with f' the non-degenerate hermitian form with the identity matrix over F. The Witt index of f equals $\frac{n}{2}$ when n is even and $\frac{n-1}{2}$ when n is odd.

5.1.2 Alternating forms

We suppose that f is a non-degenerate alternating bilinear form on a vector space V over a finite field F. As already mentioned in Section 3.1, non-degenerate alternating forms only exist on even dimensional vector spaces. Restricting to a two dimensional vector space, it is clear immediately that the Gram matrix of f must be

$$\begin{pmatrix} 0 & r \\ -r & 0 \end{pmatrix}$$

for some non-zero $r \in F$. If we rescale one of the basisvectors, which induces an isometry, then we see that there always exists a basis such that $r = 1$. We call a two dimensional vector space equipped with a non-degenerate alternating form a *symplectic hyperbolic line*, and it is proved (see Theorem 6.7 of [Cam00]) that the formed space (V, f) can be written as an orthogonal direct sum of symplectic hyperbolic planes. Hence, up to isometry, there is only one non-degenerate alternating form of an even dimensional vector space, and we choose as canonical representative the alternating form with Gram matrix

$$\begin{pmatrix} 0 & 1 & 0 & 0 & \dots & 0 & 0 \\ -1 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 & 0 \\ 0 & 0 & -1 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 0 & 1 \\ 0 & 0 & 0 & 0 & \dots & -1 & 0 \end{pmatrix}.$$

The Witt index of f equals $\frac{n}{2}$.

5.1.3 Bilinear forms

We suppose that f is a non-degenerate symmetric bilinear form on a vector space V over a finite field F with odd characteristic. We call a two dimensional vector space a *hyperbolic line* if it contains a non-zero vector v such that $f(v, v) = 0$. It is proved (see Proposition 6.9 of [Cam00]) that any two

hyperbolic lines are isometric, and we choose as canonical representative the orthogonal form with Gram matrix

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

It can be proved (see Theorem 6.10 of [Cam00]) that the formed space (V, f) can be written as the orthogonal direct sum of hyperbolic lines and one subspace U of dimension at most two. The behaviour of f on the subspace U determines the similarity class of f. We describe the three occurring cases, to describe the chosen canonical form, we use the polynomial rather than the Gram matrix.

- If the dimension of U is zero, then (V, f) is the orthogonal direct sum of hyperbolic lines, and hence (V, f) is isometric to the formed space (V, f'), where the Gram matrix of f' consists of blocks as described above. The chosen canonical form has polynomial

$$x_1x_2 + \ldots + x_{n-1}x_n$$

$$x_1x_2 + \ldots + x_{n-1}x_n$$

  Note that the dimension of the vector space V is necessarily even. We call f *hyperbolic* (see also Section 3.1). It follows also that in this similarity class, there is only one isometry class. The Witt index of f equals $\frac{n}{2}$.

- If the dimension of U is one, then necessarily the polynomial of f equals

$$\mu x_1^2 + x_2x_3 + \ldots + x_{n-1}x_n$$

  for some $\mu \in F$, and the dimension of the vector space V is odd. We call f *parabolic* (see also Section 3.1). It is clear that if μ is a square in F, then rescaling the first basis vector yields a polynomial

$$x_1^2 + x_2x_3 + \ldots + x_{n-1}x_n$$

  which we choose as the canonical form for this similarity class. If μ is a non-square, a rescaling of $x_2, x_4, \ldots, x_{n-1}$ yields a polynomial

$$\mu(x_1^2 + x_2x_3 + \ldots + x_{n-1}x_n)$$

  which represents now a bilinear form that is SIMILAR BUT NOT ISOMETRIC to the given one. Hence, the parabolic similarity class contains two isometry classes. The Witt index of f equals $\frac{n-1}{2}$.

- Suppose at last that the dimension of U is two. We may suppose that U is not a hyperbolic line. It is not too difficult to see that a suitable base change yields the polynomial

$$\mu x_1^2 + x_2^2 + x_3x_4 + \ldots + x_{n-1}x_n$$

  for a non-square $\mu \in F$, and the dimension of the vector space V is even. We call f *elliptic*. The Witt index of f equals $\frac{n-2}{2}$.

5.1.4 Degenerate forms

Suppose that f is a degenerate sesquilinear form on the vector space V, then $\mathrm{Rad}(f)$ is a non-trivial subspace of the vector space V. The vector space V can be written as the orthogonal direct sum of a subspace W and $\mathrm{Rad}(f)$, and the restriction of f to W is a non-degenerate sesquilinear form on W. Hence, f is isometric with a sesquilinear form having Gram matrix

$$\begin{pmatrix} M & A \\ B & C \end{pmatrix}$$

where M is the Gram matrix of a non-degenerate sesquilinear form and A, B and C are appropriate zero matrices. As explained in Section 3.1, the form f induces a non-degenerate form g on the vector space $V/\mathrm{Rad}(f)$. The computed matrix M can be taken as Gram matrix for the form g. As defined in Section 3.1, the Witt index of the degenerate form f is the Witt index of the non-degenerate induced-form g. The dimension of the maximal isotropic subspaces with relation to f is the sum of the Witt index and the dimension of the radical.

5.2 Morphisms of quadratic forms

Consider two formed vector spaces (V, f) and (W, g) over the same field F, where both f and g are quadratic forms. Suppose that ϕ is a linear map from V to W. The map ϕ is an *isometry* from the formed space (V, f) to the formed space (W, g) if for all v, w in V we have

$$f(v) = f'(\phi(v)).$$

The map ϕ is a *similarity* from the formed space (V, f) to a formed space (W, g) if for all v, w in V we have

$$f(v) = \lambda f'(\phi(v)).$$

for some non-zero $\lambda \in F$. Finally, the map ϕ is a *semi-similarity* from the formed space (V, f) to the formed space (W, g) if for all v, w in V we have

$$f(v) = \lambda f'(\phi(v))^{\alpha}$$

for some non-zero $\lambda \in F$ and a field automorphism α of F.

Also in this case, one of the objectives of studying maps between formed vector spaces is the classification of quadratic forms of the same vector space V, where it is sufficient to classify non-singular forms.

Since there is a one-to-one relationship between quadratic forms in odd characteristic and orthogonal bilinear forms in odd characteristic, we suppose in this section that f is a quadratic form in even characteristic. We call a two dimensional vector space a *hyperbolic line* if it contains a non-zero vector such that $f(v) = 0$. It is proved (see Proposition 6.9 of [Cam00]) that any two hyperbolic lines are isometric, and we choose as canonical representative the quadratic form with polynomial $x_1 x_2$. As in the case of the orthogonal bilinear forms, it can be proved (see Theorem 6.10 of [Cam00]) that (V, f) can be written as the orthogonal direct sum of hyperbolic lines and one subspace U of dimension at most two. The behaviour of f on the subspace U determines the similarity class of f. The classification of quadratic forms in even characteristic is analogous to the one in odd characteristic.

- If the dimension of U is zero, then (V, f) is the orthogonal direct sum of hyperbolic lines, and hence (V, f) is isometric to the formed space (V, f'), with polynomial

  $$x_1x_2 + \ldots + x_{n-1}x_n,$$

  which is chosen as the canonical form. Note that the dimension of the vector space V is necessarily even. We call f *hyperbolic* (see also Section 3.1). It follows also that in this similarity class, there is only one isometry class. The Witt index of f equals $\frac{n}{2}$.

- If the dimension of U is one, then necessarily the polynomial of f equals

  $$\mu x_1^2 + x_2x_3 + \ldots + x_{n-1}x_n$$

  for some $\mu \in F$, and the dimension of the vector space V is odd. We call f *parabolic* (see also Section 3.1). Since every element is a square in even characteristic, rescaling the first basis vector yields $\mu = 1$ The Witt index of f equals $\frac{n-1}{2}$.

- Suppose at last that the dimension of U is two. We may suppose that U is not a hyperbolic line. It is not difficult to see that a suitable base change yields the polynomial

  $$dx_1^2 + x_1x_2 + x_2^2 + x_3x_4 + \ldots + x_{n-1}x_n$$

  for an element of category 1, this is, an element d such that $\mathrm{Tr}(d) = 1$ with Tr the trace map from F to $\mathrm{GF}(2)$. Furthermore, an easy argument shows that an appropriate base change allows to choose any element of category 1 for d. It follows also that the dimension of the vector space V is even. We call f *elliptic* (see also Section 3.1). The Witt index of f equals $\frac{n-2}{2}$.

Hence, non-singular quadratic forms in even characteristic come in three similarity classes, which is analogous to the odd characteristic case, and each similarity class contains only one isometry class, which is different than in the odd characteristic case

Suppose that f is a singular quadratic form on the n-dimensional vector space V, then $\mathrm{Rad}(f)$ is a non-trivial subspace of the vector space V. The vector space V can be written as the orthogonal direct sum of a subspace W and $\mathrm{Rad}(f)$, and the restriction of f to W is a non-singular quadratic form on W. Hence, f is isometric with a quadratic form with one of the three above polynomials. The dimension of the maximal isotropic subspaces is the sum of the Witt index and the dimension of the radical.

5.2.1 Singular forms

Suppose that f is a singular quadratic form on the vector space V, then $\mathrm{Rad}(f)$ is a non-trivial subspace of the vector space V. The vector space V can be written as the orthogonal direct sum of a subspace W and $\mathrm{Rad}(f)$, and the restriction of f to W is a non-singular quadratic form on W. Hence, f is isometric with a quadratic form having Gram matrix

$$\begin{pmatrix} M & A \\ B & C \end{pmatrix}$$

where M is the Gram matrix of a non-singular quadratic form and A, B and C are appropriate zero matrices. As explained in Section 3.2, the form f induces a non-singular form g on the vector space $V/\mathrm{Rad}(f)$. The computed matrix M can be taken as Gram matrix for the form g. As defined in Section 3.2, the Witt index of the singular form f is the Witt index of the non-singular induced form g. The dimension of the maximal isotropic subspaces with relation to f is the sum of the Witt index and the dimension of the radical.

5.3 Operations based on morphisms of forms

5.3.1 BaseChangeToCanonical

▷ `BaseChangeToCanonical(f)` (attribute)

Returns: a transition matrix b from one basis to another

The argument `f` is a sesquilinear or quadratic form. For every isometry class of forms, there is a canonical representative, as described in Section 5.1. If M is the Gram matrix of the form `f`, then this method returns an invertible matrix b such that bM `TransposedMat`(b) (or bM `TransposedFrobeniusMat`(b,q) for suitable q if `f` is a hermitian form) is the Gram matrix of the canonical representative. That is, b is the *transition matrix* from a basis of the underlying vector space of `f` to another basis.

Example

```
gap> gf := GF(3);
GF(3)
gap> gram := [
> [0,0,0,1,0,0],
> [0,0,0,0,1,0],
> [0,0,0,0,0,1],
> [-1,0,0,0,0,0],
> [0,-1,0,0,0,0],
> [0,0,-1,0,0,0]] * One(gf);;
gap> form := BilinearFormByMatrix( gram, gf );
< bilinear form >
gap> b := BaseChangeToCanonical( form );;
gap> Display( b * gram * TransposedMat(b) );
 . 1 . . . .
 2 . . . . .
 . . . 1 . .
 . . 2 . . .
 . . . . . 1
 . . . . 2 .
```

5.3.2 BaseChangeHomomorphism

▷ `BaseChangeHomomorphism(b, gf)` (operation)

Returns: the inner automorphism of $\mathrm{GL}(d,q)$ associated to the transition matrix `b`.

The argument `b` must be an invertible matrix of size d over the finite field `gf` of order q. This method returns the inner automorphism of $\mathrm{GL}(d,q)$ induces by conjugation by b.

Example

```
gap> gl:=GL(3,3);
GL(3,3)
gap> go:=GO(3,3);
GO(0,3,3)
gap> form := PreservedSesquilinearForms(go)[1];
< bilinear form >
gap> gram := GramMatrix( form );
[ [ 0*Z(3), Z(3), 0*Z(3) ], [ Z(3), 0*Z(3), 0*Z(3) ],
  [ 0*Z(3), 0*Z(3), Z(3)^0 ] ]
gap> b := BaseChangeToCanonical(form);;
gap> hom := BaseChangeHomomorphism(b, GF(3));
```

```
^[ [ 0*Z(3), Z(3)^0, 0*Z(3) ], [ Z(3), Z(3), Z(3)^0 ],
  [ Z(3)^0, Z(3), 0*Z(3) ] ]
gap> newgo := Image(hom, go);
Group(
[
  [ [ Z(3)^0, Z(3)^0, 0*Z(3) ], [ 0*Z(3), Z(3), 0*Z(3) ],
      [ Z(3), Z(3)^0, Z(3) ] ],
  [ [ Z(3)^0, Z(3), 0*Z(3) ], [ Z(3), Z(3), Z(3)^0 ],
      [ 0*Z(3), Z(3)^0, 0*Z(3) ] ] ])
gap> gens := GeneratorsOfGroup(newgo);;
gap> canonical := b * gram * TransposedMat(b);
[ [ Z(3)^0, 0*Z(3), 0*Z(3) ], [ 0*Z(3), 0*Z(3), Z(3) ],
  [ 0*Z(3), Z(3), 0*Z(3) ] ]
gap> ForAll(gens, y -> y * canonical * TransposedMat(y) = canonical);
true
```

5.3.3 IsometricCanonicalForm

▷ `IsometricCanonicalForm(f)` (attribute)

Returns: the canonical form isometric to the sesquilinear or quadratic form `f`.

The argument `f` is a sesquilinear or quadratic form. For every isometry class of forms, there is a canonical representative, as described in Section 5.1, which is the returned form.

Example

```
gap> mat := [ [ Z(8) , 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2) ],
> [ 0*Z(2), Z(2)^0, Z(2^3)^5, 0*Z(2), 0*Z(2) ],
> [ 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2) ],
> [ 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2), Z(2)^0 ],
> [ 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2), 0*Z(2) ] ];;
gap> form := QuadraticFormByMatrix(mat,GF(8));
< quadratic form >
gap> iso := IsometricCanonicalForm(form);
< parabolic quadratic form >
gap> Display(form);
Parabolic quadratic form
Gram Matrix:
z = Z(8)
 z^1   .   .   .   .
   .   1 z^5   .   .
   .   .   .   .   .
   .   .   .   .   1
   .   .   .   .   .
Witt Index: 2
gap> Display(iso);
Parabolic quadratic form
Gram Matrix:
 1 . . . .
 . . 1 . .
 . . . . .
 . . . . 1
 . . . . .
Witt Index: 2
```

5.3.4 ScalarOfSimilarity

▷ `ScalarOfSimilarity(M, form)` (operation)

Returns: a finite field element

Recall that a similarity of a form f on a vector space V, is a linear transformation g of V where there exists some nonzero scalar c such that for all v, w in V,

$$f(u^g, v^g) = cf(u, v).$$

This operation finds for a particular matrix `M`, giving rise to a similarity of the sesquilinear form `form`, the said scalar c.

Example

```
gap> gram := [ [ 0*Z(3), Z(3)^0, 0*Z(3), 0*Z(3), 0*Z(3), 0*Z(3) ],
>   [ Z(3)^0, 0*Z(3), 0*Z(3), 0*Z(3), 0*Z(3), 0*Z(3) ],
>   [ 0*Z(3), 0*Z(3), 0*Z(3), Z(3)^0, 0*Z(3), 0*Z(3) ],
>   [ 0*Z(3), 0*Z(3), Z(3)^0, 0*Z(3), 0*Z(3), 0*Z(3) ],
>   [ 0*Z(3), 0*Z(3), 0*Z(3), 0*Z(3), 0*Z(3), Z(3)^0 ],
>   [ 0*Z(3), 0*Z(3), 0*Z(3), 0*Z(3), Z(3)^0, 0*Z(3) ] ];;
gap> form := BilinearFormByMatrix( gram, GF(3) );
< bilinear form >
gap> m := [ [ Z(3)^0, Z(3)^0, Z(3), 0*Z(3), Z(3)^0, Z(3) ],
>   [ Z(3), Z(3), Z(3)^0, 0*Z(3), Z(3)^0, Z(3) ],
>   [ 0*Z(3), Z(3), 0*Z(3), Z(3), 0*Z(3), 0*Z(3) ],
>   [ 0*Z(3), Z(3), Z(3)^0, Z(3), Z(3), Z(3) ],
>   [ Z(3)^0, Z(3)^0, Z(3), Z(3), Z(3)^0, Z(3)^0 ],
>   [ Z(3)^0, 0*Z(3), Z(3), Z(3)^0, Z(3), Z(3) ] ];;
gap> ScalarOfSimilarity( m, form );
Z(3)
```

5.3.5 WittIndex

▷ `WittIndex(f)` (attribute)

Returns: the Witt index of the form `f`.

The argument `f` is a sesquilinear or quadratic form on the vector space V. When `f` is degenerate, respectively singular, its Witt index is defined as the Witt index of the induced non-degenerate, respectively non-singular form on the vector space $V/\mathrm{Rad}(f)$, see Sections 3.1 and 3.2.

Example

```
gap> mat := [[0,0,1,0,0],[0,0,0,0,0],[-1,0,0,0,0],[0,0,0,0,0],[0,0,0,0,0]]*Z(7)^0;
[ [ 0*Z(7), 0*Z(7), Z(7)^0, 0*Z(7), 0*Z(7) ],
  [ 0*Z(7), 0*Z(7), 0*Z(7), 0*Z(7), 0*Z(7) ],
  [ Z(7)^3, 0*Z(7), 0*Z(7), 0*Z(7), 0*Z(7) ],
  [ 0*Z(7), 0*Z(7), 0*Z(7), 0*Z(7), 0*Z(7) ],
  [ 0*Z(7), 0*Z(7), 0*Z(7), 0*Z(7), 0*Z(7) ] ]
gap> form := BilinearFormByMatrix(mat,GF(7));
< bilinear form >
gap> WittIndex(form);
1
gap> RadicalOfForm(form);
<vector space over GF(7), with 3 generators>
gap> Dimension(last);
```

```
3
gap> mat := IdentityMat(6,GF(5));
[ [ Z(5)^0, 0*Z(5), 0*Z(5), 0*Z(5), 0*Z(5), 0*Z(5) ],
  [ 0*Z(5), Z(5)^0, 0*Z(5), 0*Z(5), 0*Z(5), 0*Z(5) ],
  [ 0*Z(5), 0*Z(5), Z(5)^0, 0*Z(5), 0*Z(5), 0*Z(5) ],
  [ 0*Z(5), 0*Z(5), 0*Z(5), Z(5)^0, 0*Z(5), 0*Z(5) ],
  [ 0*Z(5), 0*Z(5), 0*Z(5), 0*Z(5), Z(5)^0, 0*Z(5) ],
  [ 0*Z(5), 0*Z(5), 0*Z(5), 0*Z(5), 0*Z(5), Z(5)^0 ] ]
gap> form := QuadraticFormByMatrix(mat,GF(5));
< quadratic form >
gap> WittIndex(form);
3
gap> mat := IdentityMat(6,GF(7));
[ [ Z(7)^0, 0*Z(7), 0*Z(7), 0*Z(7), 0*Z(7), 0*Z(7) ],
  [ 0*Z(7), Z(7)^0, 0*Z(7), 0*Z(7), 0*Z(7), 0*Z(7) ],
  [ 0*Z(7), 0*Z(7), Z(7)^0, 0*Z(7), 0*Z(7), 0*Z(7) ],
  [ 0*Z(7), 0*Z(7), 0*Z(7), Z(7)^0, 0*Z(7), 0*Z(7) ],
  [ 0*Z(7), 0*Z(7), 0*Z(7), 0*Z(7), Z(7)^0, 0*Z(7) ],
  [ 0*Z(7), 0*Z(7), 0*Z(7), 0*Z(7), 0*Z(7), Z(7)^0 ] ]
gap> form := QuadraticFormByMatrix(mat,GF(7));
< quadratic form >
gap> WittIndex(form);
2
```

5.3.6 IsEllipticForm

▷ `IsEllipticForm(f)` (property)

Returns: true or false.

The argument `f` is a sesquilinear or quadratic form on the vector space V. This operation returns `true` is and only if `f` is elliptic; that is, it is orthogonal of minus type, or in other words, has even dimension and non-maximal Witt index (see Section 5.1.3 for sesquilinear forms and Section 5.2 for quadratic forms). If `f` is degenerate, respectively singular, then this operation refers to the inuced non-degenerate, respectively non-singular form induced on the vector space $V/\mathrm{Rad}(f)$.

5.3.7 IsParabolicForm

▷ `IsParabolicForm(f)` (property)

Returns: true or false.

The argument `f` is a sesquilinear or quadratic form on the vector space V. This operation returns `true` is and only if `f` is parabolic; that is, it is orthogonal of neutral type, or in other words, it has odd dimension (see Section 5.1.3 for sesquilinear forms and Section 5.2 for quadratic forms). If `f` is degenerate, respectively singular, then this operation refers to the inuced non-degenerate, respectively non-singular form induced on the vector space $V/\mathrm{Rad}(f)$.

5.3.8 IsHyperbolicForm

▷ `IsHyperbolicForm(f)` (attribute)

Returns: true or false.

The argument `f` is a sesquilinear or quadratic form on the vector space V. This operation returns `true` is and only if `f` is hyperbolic; that is, it is orthogonal of plus type, or in other words, has even dimension and maximal Witt index (see Section 5.1.3 for sesquilinear forms and Section 5.2 for quadratic forms). If `f` is degenerate, respectively singular, then this operation refers to the inuced non-degenerate, respectively non-singular form induced on the vector space $V/\mathrm{Rad}(f)$.

5.3.9 TypeOfForm

▷ `TypeOfForm(f)` (operation)

Returns: a number.

The argument `f` is a sesquilinear or quadratic form on the vector space V with radical R, a k-dimensional space. Then `f` induces a non-degenerate/non-singular form g on V/R. When R is the trivial vector space, the form g is just the given form `f`. This opertion returns

- 0 when g is symplecitc or parabolic;
- +1 when g is hyperbolic;
- -1 when g is elliptic;
- -1/2 when g is hermitian in odd dimension;
- +1/2 when g is hermitian in even dimension;
- an error message when `f` is a pseudo form.

Note that no method is installed for the trivial form. The methods for this operation rely on `IsParabolicForm`, `IsHyperbolicForm` and `IsEllipticForm` for orthogonal bilinear forms and quadratic forms.

Example

```
gap> mat := [[0,0,0,-1],[0,0,3,0],[0,-3,0,0],[1,0,0,0]]*Z(25)^0;
[ [ 0*Z(5), 0*Z(5), 0*Z(5), Z(5)^2 ], [ 0*Z(5), 0*Z(5), Z(5)^3, 0*Z(5) ],
  [ 0*Z(5), Z(5), 0*Z(5), 0*Z(5) ], [ Z(5)^0, 0*Z(5), 0*Z(5), 0*Z(5) ] ]
gap> form := BilinearFormByMatrix(mat,GF(25));
< bilinear form >
gap> IsDegenerateForm(form);
false
gap> TypeOfForm(form);
0
gap> mat := IdentityMat(3,GF(7));
[ [ Z(7)^0, 0*Z(7), 0*Z(7) ], [ 0*Z(7), Z(7)^0, 0*Z(7) ],
  [ 0*Z(7), 0*Z(7), Z(7)^0 ] ]
gap> form := QuadraticFormByMatrix(mat,GF(7));
< quadratic form >
gap> IsSingularForm(form);
false
gap> TypeOfForm(form);
0
gap> mat := [[0,1,0,0],[-1,0,0,0],[0,0,0,0],[0,0,0,0]]*Z(5)^0;
[ [ 0*Z(5), Z(5)^0, 0*Z(5), 0*Z(5) ], [ Z(5)^2, 0*Z(5), 0*Z(5), 0*Z(5) ],
  [ 0*Z(5), 0*Z(5), 0*Z(5), 0*Z(5) ], [ 0*Z(5), 0*Z(5), 0*Z(5), 0*Z(5) ] ]
gap> form := BilinearFormByMatrix(mat,GF(5));
```

```
< bilinear form >
gap> IsDegenerateForm(form);
true
gap> TypeOfForm(form);
0
gap> mat := [[1,0,0,0],[0,1,0,0],[0,0,0,1],[0,0,1,0]]*Z(7)^0;
[ [ Z(7)^0, 0*Z(7), 0*Z(7), 0*Z(7) ], [ 0*Z(7), Z(7)^0, 0*Z(7), 0*Z(7) ],
  [ 0*Z(7), 0*Z(7), 0*Z(7), Z(7)^0 ], [ 0*Z(7), 0*Z(7), Z(7)^0, 0*Z(7) ] ]
gap> form := BilinearFormByMatrix(mat,GF(7));
< bilinear form >
gap> IsDegenerateForm(form);
false
gap> TypeOfForm(form);
-1
gap> mat := IdentityMat(3,GF(9));
[ [ Z(3)^0, 0*Z(3), 0*Z(3) ], [ 0*Z(3), Z(3)^0, 0*Z(3) ],
  [ 0*Z(3), 0*Z(3), Z(3)^0 ] ]
gap> form := HermitianFormByMatrix(mat,GF(9));
< hermitian form >
gap> IsDegenerateForm(form);
false
gap> TypeOfForm(form);
-1/2
gap> mat := [[0,0,0,1],[0,1,0,0],[0,0,1,0],[1,0,0,0]]*Z(8)^0;
[ [ 0*Z(2), 0*Z(2), 0*Z(2), Z(2)^0 ], [ 0*Z(2), Z(2)^0, 0*Z(2), 0*Z(2) ],
  [ 0*Z(2), 0*Z(2), Z(2)^0, 0*Z(2) ], [ Z(2)^0, 0*Z(2), 0*Z(2), 0*Z(2) ] ]
gap> form := BilinearFormByMatrix(mat,GF(8));
< bilinear form >
gap> IsDegenerateForm(form);
false
gap> TypeOfForm(form);
Error, <f> is a pseudo form and has no defined type called from
<function "unknown">( <arguments> )
 called from read-eval loop at line 30 of *stdin*
you can 'quit;' to quit to outer loop, or
you can 'return;' to continue
brk> quit;
```

References

[Asc00] M. Aschbacher. *Finite group theory*, volume 10 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, Cambridge, second edition, 2000. 10, 41

[Cam00] Peter J. Cameron. *Projective and Polar Spaces*. Online notes, http://www.maths.qmul.ac.uk/~pjc/pps, 2000. 10, 41, 42, 43, 44

[CCN+85] J. H. Conway, R. T. Curtis, S. P. Norton, R. A. Parker, and R. A. Wilson. *Atlas of finite groups*. Oxford University Press, Eynsham, 1985. Maximal subgroups and ordinary characters for simple groups, With computational assistance from J. G. Thackray. 10, 11, 16, 36

[CLGN+08] Frank Celler, Charles R. Leedham-Green, Peter M. Neumann, Alice C. Niemeyer, Dmitri Pasechnik, and Cheryl E. Praeger. Finding invariant forms. *preprint*, 2008. 36

[KL90] Peter Kleidman and Martin Liebeck. *The subgroup structure of the finite classical groups*, volume 129 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, Cambridge, 1990. 10, 41, 42

[Tay92] Donald E. Taylor. *The geometry of the classical groups*, volume 9 of *Sigma Series in Pure Mathematics*. Heldermann Verlag, Berlin, 1992. 10, 11, 41

Index