Conjugacy Classes in Finite Orthogonal Groups

D. E. Taylor

Version of 17 March, 2016
TEXed: 10 October, 2018

The definitive and very general treatment of the conjugacy classes of the unitary, symplectic and orthogonal groups was given by Wall [17] in 1963, building on the work of Williamson [18] for perfect fields of characteristic other than two.

The following sections describe Magma code [1, 2] implementing the construction of conjugacy classes for the special case of the finite orthogonal groups defined over a Galois field GF(q), where q is odd. The approach given here follows Milnor [12] combined with the work of Wall [17] as interpreted by Fulman [4, Chapter 6]. For other approaches with more emphasis on the theory of algebraic groups see Springer-Steinberg [15], Humphreys [8] and the recent book of Liebeck and Seitz [10]. For fields of characteristic two see Hesselink [7] and Xue [20].

The conjugacy classes are obtained by first computing a complete collection of invariants and then determining a representative matrix for each invariant.

A partial analysis of similar algorithms for finite unitary groups can be found in [6]. There are some remarks about the orthogonal groups in the unpublished draft [13].

1 Orthogonal groups

In Magma the general orthogonal group in dimension n over the field $k = \mathrm{GF}(q)$, where q is odd, consists of $n \times n$ matrices A over k such that $AJA^{\mathrm{tr}} = J$, where A^{tr} denotes the transpose of A and J is a 'standard' non-degenerate symmetric matrix. For all n there are two isometry classes of symmetric bilinear forms. However if n is odd, up to isomorphism, there is just one orthogonal group, whereas if n is even there are two orthogonal groups.

If $V = k^n$, the matrix J defines a symmetric bilinear form on V given by $\beta(u, v) = uJv^{\mathrm{tr}}$.

Odd dimensional orthogonal groups, q odd

In $n = 2m + 1$, every non-degenerate symmetric bilinear form is congruent to a form with matrix

$$J = \begin{pmatrix} 0 & 0 & \Lambda_m \\ 0 & a & 0 \\ \Lambda_m & 0 & 0 \end{pmatrix} \quad \text{where } \Lambda_m \text{ is the } m \times m \text{ matrix} \quad \begin{pmatrix} 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & \cdots & 1 & 0 \\ & & \iddots & & \\ 0 & 1 & \cdots & 0 & 0 \\ 1 & 0 & \cdots & 0 & 0 \end{pmatrix}$$

and where a is a non-zero element of k. There are two congruence classes of these forms determined by whether or not a is a square in k. In the following code we use the forms with $a = 1$ and $a = \delta$, where δ is a fixed non-square in k. In MAGMA, the default return value of STANDARDSYMMETRICFORM(n, q) has $a = 2$ whereas the symmetric bilinear form preserved by GO(n, q) has $a = 1/2$; this is the form returned when the `Variant := "Revised"` option is used.

Even dimensional orthogonal groups, q odd

The group $\mathrm{GO}^+(2m, q)$ consists of the matrices A such that $AJA^{\mathrm{tr}} = J$, where $J = \begin{pmatrix} 0 & \Lambda_m \\ \Lambda_m & 0 \end{pmatrix}$ is the matrix returned by STANDARDSYMMETRICFORM(n, q).

The group $\mathrm{GO}^-(2m+2, q)$ consists of the matrices A such that $AJA^{\mathrm{tr}} = J$, where $J = \begin{pmatrix} 0 & 0 & \Lambda_m \\ 0 & J_2 & 0 \\ \Lambda_m & 0 & 0 \end{pmatrix}$, $J_2 = \begin{pmatrix} 1 & 0 \\ 0 & -\delta \end{pmatrix}$ and, as above, δ is a non-square in k. This is returned by the MAGMA command STANDARDSYMMETRICFORM($2{*}m{+}2, q$: VARIANT := REVISED, MINUS).

Conjugacy and modules

$\mathrm{GO}^\varepsilon(V)$ denotes an orthogonal group on $V = k^q$, where $\varepsilon \in \{0, +, -\}$ and where $\mathrm{GO}^0(V) = \mathrm{GO}(V)$. The description of the conjugacy classes of $\mathrm{GO}^\varepsilon(V)$ closely parallels the description of the conjugacy classes of $\mathrm{GL}(V)$.

For $g \in \mathrm{GL}(V)$, the space V becomes a $k[t]$-module V_g by defining $vf(t) = vf(g)$ for all $v \in V$ and all $f(t) \in k[t]$.

The symmetric form β defines an isomorphism $\theta : V \to V^* : v \mapsto \beta(-, v)$. An element $g \in \mathrm{GL}(V)$ acts on V^* according to the rule $v(\psi g) = (vg^{-1})\psi$ for all $v \in V$ and all $\psi \in V^*$; that is, $\psi g = g^{-1}\psi$. With this action V^* becomes a $k[t]$-module V_g^* and g belongs to $\mathrm{GO}^\varepsilon(V)$ if and only if $\theta : V_g \to V_g^*$ is an isomorphism of $k[t]$-modules.

If $g, h \in \mathrm{GL}(V)$, then $T : V_g \to V_h$ is a $k[t]$-isomorphism if and only if $gT = Th$ if and only if $T^{-1}g^{-1} = h^{-1}T^{-1}$ if and only if $T : V_g^* \to V_h^*$ is an isomorphism. Since $T \in \mathrm{GO}^\varepsilon(V)$ if and only if $T\theta = \theta T$ it follows that $g, h \in \mathrm{GO}^\varepsilon(V)$ are conjugate in $\mathrm{GO}^\varepsilon(V)$ if and only if there is a $k[t]$-isomorphism $T : V_g \to V_h$ such that the diagram

$$\begin{tikzcd} V_g \arrow[r, "T"] \arrow[d, "\theta"'] & V_h \arrow[d, "\theta"] \\ V_g^* \arrow[r, "T"'] & V_h^* \end{tikzcd}$$

commutes.

As shown in Macdonald [11, Chap. IV], if $\mathcal{P}$ is the set of all partitions and Φ is the set of all monic irreducible polynomials (other than t), then for $g \in \mathrm{GL}(V)$ there is a function $\mu : \Phi \to \mathcal{P}$ such that

$$V_g = \bigoplus_{f \in \Phi, i} k[t]/(f)^{\mu_i(f)} \tag{1.1}$$

and $\mu(f) = (\mu_1(f), \mu_2(f), \dots,)$ is a partition such that

$$\sum_{f \in \Phi} \deg(f)|\mu(f)| = n.$$

For $g \in \mathrm{GO}^\varepsilon(V)$ there are restrictions—to be determined in the sections which follow—on the polynomials and partitions that can occur in this decomposition.

The functions listed in the following **import** statement were defined in `SpConjugacy.tex` and written to the file `common.m`. For clarity of exposition some of this the code will be repeated below coloured red, but not written to the primary MAGMA file `GOConjugacy.m`.

import "`common.m`" : *convert*, *allPartitions*, *primaryParts*, *stdJordanBlock*, *centralJoin*, *getSubIndices*, *restriction*, *homocyclicSplit*, *type1Companion*;

Polynomials

Definition 1.1.

(i) If $f(t) \in k[t]$ is a monic polynomial of degree d such that $f(0) \neq 0$, the *dual* of $f(t)$ is the polynomial
$$f^*(t) = f(0)^{-1} t^d f(t^{-1}).$$

(ii) The polynomial $f(t) \neq t$ is *$*$-symmetric* if $f^*(t) = f(t)$.

(iii) A polynomial $f(t)$ is *$*$-irreducible* if it is $*$-symmetric and has no proper $*$-symmetric factors.

The monic polynomial $f(t) = a_0 + a_1 t + \cdots + a_{d-1} t^{d-1} + t^d$ is $*$-symmetric if and only if
$$a_0^2 = 1 \quad \text{and} \quad a_{d-i} = a_0 a_i \quad \text{for } 0 < i < d. \tag{1.2}$$
Hence $a_0 = \pm 1$. Furthermore, an element a in an extension field of k is a root of a $*$-symmetric polynomial $f(t)$ if and only if a^{-1} is also a root.

If g preserves the symmetric form β introduced above, then for all $u, v \in V$ we have
$$\beta(ug, v) = \beta(u, vg^{-1})$$
and thus for $f(t) \in k[t]$ we have
$$\beta(uf(g), v) = \beta(u, vf(g^{-1})). \tag{1.3}$$
In particular, if $m(t)$ is the minimal polynomial of g, then $vm(g^{-1}) = 0$ for all v and therefore $g^d m(g^{-1}) = 0$, where d is the degree of $m(t)$. Thus $m^*(g) = 0$ and it follows that $m^*(t) = m(t)$; that is, the minimal polynomial of g is $*$-symmetric.

Lemma 1.2. *Let $f(t)$ be a monic $*$-irreducible polynomial.*

(i) *If $f(t)$ is reducible, there exists an irreducible polynomial $g(t)$ such that $f(t) = g(t)g^*(t)$ and $g(t) \neq g^*(t)$.*

(ii) *If the degree of $f(t)$ is even, then $f(0) = 1$.*

(iii) *If $f(t)$ is irreducible and of odd degree, then $f(t)$ is either $t - 1$ or $t + 1$.*

(iv) *If $f(t)$ is irreducible of even degree $2d$, there is an irreducible polynomial $g(t)$ of degree d such that $f(t) = t^d g(t + t^{-1})$.*

Proof. (i) Suppose that $g(t)$ is an irreducible factor of $f(t)$. Then $g^*(t)$ divides $f^*(t) = f(t)$ and since $f(t)$ is $*$-irreducible $f(t) = g(t)g^*(t)$ or $f(t) = g(t)$.

(ii) Suppose that the degree of $f(t)$ is $2d$. We may suppose that the characteristic of the field is not 2. If $a_0 = -1$ it follows from (1.2) that $a_d = 0$ and that $a_{2d-i} = -a_i$ for $1 \le i < d$. Thus $f(1) = 0$ and so $t - 1$ divides $f(t)$. If $f(t) = g(t)g^*(t)$ and $g(t)$ is irreducible, as in (i), then $t - 1$ divides $g(t)$, whence $g(t) = t - 1$ and thus $g(t) = g^*(t)$, contradicting (i). If $f(t)$ is irreducible, then $f(t) = t - 1$ contradicting the assumption that the degree of $f(t)$ is even.

(iii) Suppose that $f(t)$ is irreducible and that its degree is odd. It follows from (1.2) that $f(-a_0) = 0$ where $a_0 = \pm 1$ is the constant term of $f(t)$. Thus $f(t) = t + a_0$, proving (iii).

(iv) Suppose that $f(t)$ is irreducible of degree $2d$. Then from (ii) we have $a_0 = 1$ and it follows by induction (successively subtracting multiples of $(t + t^{-1})^i$ from $t^{-d} f(t)$) that there exists a polynomial $g(t)$ such that $f(t) = t^d g(t + t^{-1})$. □

The intrinsic StarIrreduciblePolynomials(F, d) (defined in `SpConjugacy.m`) returns all $*$-symmetric monic polynomials over the field F of degree d with no proper $*$-symmetric factors.

Partitions

Given a partition in the form $[\lambda_1, \lambda_2, \ldots, \lambda_n]$, convert it to a sequence of multiplicities $[\langle 1, m_1 \rangle, \langle 2, m_2 \rangle, \ldots, \langle n, m_n \rangle]$, omitting the terms with $m_i = 0$.

```
convert := function(partition)
   n := Max(partition);
   mults := [ 0 : i in [1..n] ];
   for λ in partition do mults[λ] +:= 1; end for;
   return [<i, mults[i]> : i in [1..n] | mults[i] ne 0 ];
end function;
```

The function *allPartitions*(d) returns a sequence of length d whose nth term is the list of partitions of n.

```
allPartitions := func <d | [[convert(π) : π in Partitions(n)] : n in [1..d]] >;
```

Definition 1.3. An orthogonal *signed partition* is a sequence $[\langle \pm 1, m_1 \rangle, \langle 2, m_2 \rangle, \langle \pm 3, m_3 \rangle, \ldots]$ such that $[\langle 1, m_1 \rangle, \langle 2, m_2 \rangle, \langle 3, m_3 \rangle, \ldots]$ is the sequence of multiplicities of a partition, m_i is even for all even i and there is a sign associated to each pair $\langle i, m_i \rangle$ for all odd i. From now on, the terms with $m_i = 0$ will be omitted from the sequence.

Thus a signed partition π is a list of pairs $\lambda = \langle e, m \rangle$. If e is even, λ is of *symplectic* type; if e is odd, λ is of *orthogonal* type. The absolute value of e will be the exponent of an associated irreducible polynomial.

```
isSignedPartition := func< π |
   forall{ λ : λ in π | IsOdd(λ[1]) or (IsEven(λ[2]) and λ[1] gt 0) } >;

addSignsO := function(plist)
   slist := [];
   for π in plist do
      if forall{ μ : μ in π | IsOdd(μ[1]) or IsEven(μ[2])} then
         ndx := { i : i in [1..#π] | IsOdd(π[i][1]) };
```

```
            for S in SUBSETS(ndx) do
               $\lambda$ := $\pi$;
               for i in S do
                  $\mu$ := $\pi[i]$;
                  $\lambda[i]$ := < $-\mu[1], \mu[2]$ >;
               end for;
               APPEND(~slist, $\lambda$);
            end for;
         end if;
      end for;
      return slist;
   end function;

   signedPartitionsO := func< d | [ addSignsO(plist) : plist in allPartitions(d) ] >;
```

It turns out (cf. Shinoda [14]) that the conjugacy classes of $\mathrm{GO}^{\varepsilon}(V)$ are parametrised by functions $\mu : \Phi^* \to \mathcal{P} \cup \mathcal{S}$, where Φ^* is the set of (monic) $*$-irreducible polynomials and $\mathcal{S}$ is the set of signed partitions such that $\mu(f) \in \mathcal{S}$ if and only if $f(t) = t \pm 1$.

We shall refer to μ as a *conjugacy invariant* and represent it as an indexed set of pairs $\langle f, \pi \rangle$, where f is a $*$-irreducible polynomial and π is either a partition or, when f is $t+1$ or $t-1$, a signed partition. An example of a conjugacy invariant, where $k = \mathrm{GF}(11)$, is

```
{@ <t + 1, [<2,1>]>, <t^4 + 7t^3 + 7t + 1, [< −1,2>, <2,1>]> @}.
```

1.1 Quadratic spaces and the Witt ring of GF(q), q odd

A *quadratic space* is a pair (V, Q), where V is a vector space over a field k and $Q : V \to k$ is a quadratic form on V with polar form β. We shall suppose that Q is non-degenerate and that the field k is GF(q), where q is odd. A general reference for this section is [16, Chapter 11].

Definition 1.4.

(i) A non-zero vector v is *singular* if $Q(v) = 0$.

(ii) A pair of vectors u, v is a *hyperbolic pair* if u and v are singular and $\beta(u, v) = 1$. The subspace spanned by u and v is called a *hyperbolic plane*.

(iii) The *discriminant* dV of V is the determinant, modulo k^2, of a matrix representing β.

(iv) A quadratic space is a *metabolic* space if it is the orthogonal sum of hyperbolic planes. The discriminant of a hyperbolic plane is -1 and therefore the discriminant of a metabolic space that is the sum of m hyperbolic planes is $(-1)^m$.

Lemma 1.5. *If a and b are non-zero elements of k, then for all $c \in k$ there exist $x, y \in k$ such that $c = ax^2 + by^2$.* □

From now on, because we assume that q is odd, we regard a quadratic space as a pair (V, β) and use the notation $V = \langle a_1, a_2, \ldots, a_m \rangle$ to mean that V has an orthogonal basis v_1, $v_2, \ldots$ v_m such that $\beta(v_i, v_i) = a_i$ for all i. In particular, $\langle 0 \rangle$ is the unique quadratic space of dimension 0.

Corollary 1.6. *We have $V = \langle 1, 1, \ldots, 1, a \rangle$, where a is either 1 or a non-square in k. In this case $dV = a$.* □

Lemma 1.7. *If V is a quadratic space of dimension at least 3, then V contains a singular vector.* □

Corollary 1.8. *The quadratic space V can be written in the form $V = M \perp V_0$, where M is a metabolic space,* $\dim V_0 \leq 2$ *and there are no singular vectors in V_0.* □

The space V_0 is called the *anisotropic kernel* of V. It is uniquely determined by V up to isometry. The *Witt index* of V is $\frac{1}{2} \dim M$. The Witt index is said to be *maximal* if $V_0 = 0$.

Lemma 1.9. *Quadratic spaces of the same dimension are isometric if and only if their anisotropic kernels are isometric.* □

Definition 1.10. Two quadratic spaces are *equivalent* if their anisotropic kernels are isometric. This equivalence relation is compatible with orthogonal sum and tensor product and hence the set of equivalence classes becomes a ring $W(k)$, called the *Witt ring* of k.

The original construction of $W(k)$ is in Witt [19] and a more extended exposition can be found in Knebusch and Kolster [9] as well as in many other places.

For the finite field $k = \mathrm{GF}(q)$, the Witt ring $W(k)$ has four elements, represented by $\langle 0 \rangle$, $\langle 1 \rangle$, $\langle \delta \rangle$ and $\langle 1, -\delta \rangle$, where δ is a non-square in k. The structure of $W(k)$ depends on the congruence of q modulo 4. If $q \equiv 1 \pmod 4$, then $W(k) \simeq \mathbb{Z}_2 \oplus \mathbb{Z}_2$ whereas if $q \equiv 3 \pmod 4$, then $W(k) \simeq \mathbb{Z}_4$. If $q \equiv 3 \pmod 4$, we may take $\delta = -1$ and in this case it is clear that $\langle 1 \rangle$ and $\langle -1 \rangle$ are the elements of order 4 in $W(k)$.

Remark 1.11. Suppose that the dimension of V is $2m$. If V has maximal Witt index, the discriminant is $(-1)^m \pmod{k^2}$ whereas if the Witt index is not maximal, the discriminant is $(-1)^m \delta \pmod{k^2}$. In particular, if $m \equiv 0 \pmod 4$ or if $q \equiv 1 \pmod 4$, the discriminant is a square if and only if the Witt index is maximal.

Definition 1.12. The *sign* of a non-degenerate quadratic space V is $+1$ if its anisotropic kernel is $\langle 0 \rangle$ or $\langle 1 \rangle$; otherwise the sign is -1. Thus, if the dimension of V is even, its sign is $+1$ if and only if its Witt index is maximal.

In later sections, for every conjugacy invariant ξ, we construct a matrix $g = \rho(\xi)$, which is an isometry of a quadratic space V. Recall that a conjugacy invariant ξ is an indexed set of pairs $\langle f, \mu \rangle$ where f is a $*$-irreducible polynomial and μ is a partition, which is signed if the degree of f is 1. The partition μ can be represented as a sequence of pairs of integers $\langle e, m \rangle$. Hence ξ may also be represented as a sequence of triples $\langle f, e, m \rangle$, where f is $*$-irreducible and e and m are integers with $m > 0$. Moreover $e > 0$ if $\deg f > 1$ and m is even if $\deg f = 1$ and e is even.

We shall see that the decomposition (1.1) of V_g can be refined to an orthogonal sum of subspaces $V_{f,e,m} \simeq mk[t]/(f(t)^{|e|})$ corresponding to the triples $\langle f, e, m \rangle$. Thus e is the *exponent* and m is the *multiplicity* of the term. Let ω denote the Witt type of the anisotropic space of dimension 2. If f is reducible, the Witt type of $V_{f,e,m}$ is $\langle 0 \rangle$; if f is irreducible and $\deg f > 1$, the Witt type of $V_{f,e,m}$ is $\langle 0 \rangle$ or ω according to whether e is even or odd.

If the degree of f is 1, let $\Theta(f, e, m)$ denote the image of $V_{f,e,m}/V_{f,e,m}f(t)$ in the Witt ring. Then $\Theta(f, e, m)$ is the Witt type of a quadratic space of dimension m and whose sign is the sign of e. If e is even, we have $\Theta(f, e, m) = \langle 0 \rangle$. Hence (see [5, page 213] or [17, Lemma 2.6.1]) the image of V_g in the Witt ring is

$$\sum_{\deg(f)=1} \sum_{e \text{ odd}} \Theta(f, e, m) + \sum_{\substack{\deg(f)>1 \\ f \text{ irreducible}}} \sum_{em \text{ odd}} \omega.$$

The following functions implement this formula. The justification will be given later (see Corollary 3.9, for example). First we provide a function to convert a conjugacy invariant to an element of the Witt ring $W(k)$.

```
invToWitt := function(q, inv)
  qmod4 := q mod 4 eq 1;
```

Set up the Witt group of the field GF(q).

```
  W := AbelianGroup(GrpAb, qmod4 select [2,2] else [4]);
  w := W ! 0;
  ω := qmod4 select W.1 + W.2 else 2*W.1;
  for pol_part in inv do
    f, λ := Explode(pol_part);
    d := Degree(f);
    if IsIrreducible(f) then
      for μ in λ do
        e, m := Explode(μ);
        if d eq 1 then
```

If m is even and the sign of the quadratic space is $+1$ its Witt index is maximal and it is represented by $\langle 0 \rangle$, the zero of W; if the sign is -1, it is represented by ω.

```
          if IsEven(m) then
            if e lt 0 then w +:= ω; end if;
          else
```

If m is odd, we *choose* the quadratic space of sign $+1$ to be represented by $W.1$. Then the space with sign -1 is represented either by $W.2$ or $3W.1$ according to whether $q \equiv 1$ or $3 \pmod 4$.

```
            if e gt 0 then w +:= W.1;
            else w +:= qmod4 select W.2 else 3*W.1;
            end if;
          end if;
        else
          w +:= e*m*ω;
        end if;
      end for;
    end if;
  end for;
  return w;
end function;
```

Recall that the sign is $+1$ if the Witt type is $\langle 0 \rangle$ or $\langle 1 \rangle$, otherwise it is -1.

```
theSign := function(q, inv)
```

```
    w := invToWitt(q, inv);
    W := Parent(w);
    return w in {W ! 0, W.1} select 1 else −1;
  end function;
```

Encode the Witt type as an integer $d \in \{-1, 0, 1, 2\}$, where $|d|$ is the dimension of the anisotropic kernel and, if d is odd, the sign of d is the sign of the Witt type. We call d the Witt *code*.

```
  encodeWitt := function(q, inv)
    w := invToWitt(q, inv);
    W := Parent(w);
    return (q mod 4 eq 1)
      select case< w | W.1 : 1, W.2 : −1, W ! 0 : 0, default : 2 >
      else case< w | W.1 : 1, 3∗W.1 : −1, W ! 0 : 0, default : 2 >;
  end function;
```

The encoding of a triple $\langle f, e, m \rangle$ can be computed directly.

```
  wittCode := function(f, e, m)
    code := 0;
    if IsIrreducible(f) then
      if Degree(f) eq 1 then
        if IsEven(m) then
          if (e lt 0) then code := 2; end if;
        else
          code := (e gt 0) select 1 else −1;
        end if;
      elif IsOdd(e) and IsOdd(m) then
        code := 2;
      end if;
    end if;
    return code;
  end function;
```

1.2 Class invariants

With polynomials, partitions and the Witt ring at our disposal we can construct the class invariants for all orthogonal groups. The sequence returned by this function is the union of the invariants for *all* orthogonal groups of dimension d over the field GF(q). The particular group to which the invariant is associated depends on its sign and so, in effect, the orthogonal groups for d odd are represented twice.

```
  classInvariantsO := function(d, q)
    error if IsEven(q), "q must be odd";
    pols := &cat[ StarIrreduciblePolynomials(GF(q), i) : i in [1..d] ];
    parts := allPartitions(d);
    sparts := signedPartitionsO(d);
    oldinv := [ [] : n in [1..d] ];
    inv := oldinv;
  for $f$ in $pols$ do
    $fparts$ := (Degree($f$) eq 1) select $sparts$ else $parts$;
    for $n$ := 0 to $d-1$ do
      $dimleft$ := $d-n$;
      if Degree($f$) le $dimleft$ then
        $multleft$ := $dimleft$ div Degree($f$);
        for $i$ := 1 to $multleft$ do
          for $param$ in (($n$ ne 0) select $oldinv[n]$ else [ {@ @}]) do
            for $part$ in $fparts[i]$ do
              Append( ~$inv[n+$Degree($f$)$*i]$, Include($param$, <$f$, $part$>) );
            end for;
          end for;
        end for;
      end if;
    end for;
    $oldinv$ := $inv$;
  end for;
  return $inv[d]$;
end function;

intrinsic ClassInvariantsGO($d$ :: RngIntElt, $q$ :: RngIntElt) $\to$ SeqEnum
{The conjugacy class invariants for the orthogonal group
 GO(d,q) }
  require IsOdd($d$) : "d should be odd";
  return [ $\xi$ : $\xi$ in $classInvariantsO(d,q)$ | $theSign(q,\xi)$ eq 1 ];
end intrinsic;

intrinsic ClassInvariantsGOPlus($d$ :: RngIntElt, $q$ :: RngIntElt) $\to$ SeqEnum
{The conjugacy class invariants for the orthogonal group
 GOPlus(d,q) }
  require IsEven($d$) : "d should be even";
  return [ $\xi$ : $\xi$ in $classInvariantsO(d,q)$ | $theSign(q,\xi)$ eq 1 ];
end intrinsic;

intrinsic ClassInvariantsGOMinus($d$ :: RngIntElt, $q$ :: RngIntElt) $\to$ SeqEnum
{The conjugacy class invariants for the orthogonal group
 GOMinus(d,q) }
  require IsEven($d$) : "d should be even";
  return [ $\xi$ : $\xi$ in $classInvariantsO(d,q)$ | $theSign(q,\xi)$ eq $-1$ ];
end intrinsic;
```

2 An hermitian form

Throughout this section g is an element of $\mathrm{GO}^{\varepsilon}(V)$ whose minimal polynomial $m(t)$ is irreducible of degree d. We follow the exposition in Milnor [12, §1].

In this case V is a vector space over the field $E = k[t]/(m(t)) = k[\theta]$, where $\theta = t + (m(t))$ and the linear transformation g becomes right multiplication by θ; that is, $g : v \mapsto v\theta$.

We have already seen that $m(t)$ is $*$-symmetric and so $m(\theta^{-1}) = 0$. It follows that there is an automorphism $e \mapsto \bar{e}$ of E such that $\bar{\theta} = \theta^{-1}$. The automorphism is the identity if and only if $\theta^2 = 1$ (in which case $E \simeq k$) and so for the remainder of this section we assume that $m(t)$ is neither $t+1$ nor $t-1$. Then (1.3) becomes

$$\beta(ue, v) = \beta(u, v\bar{e}).$$

For fixed $u, v \in V$ the map $E \to k : e \mapsto \beta(ue, v)$ is k-linear and so there is a unique element $u \circ v \in E$ such that

$$\mathrm{Tr}_{E/k}(e(u \circ v)) = \beta(ue, v) \quad \text{for all } e \in E.$$

Lemma 2.1. *$u \circ v$ is the unique hermitian inner product on V such that*

$$\beta(u, v) = \mathrm{Tr}_{E/k}(u \circ v).$$

Moreover $u \circ v$ is non-degenerate.

Proof. By definition

$$\mathrm{Tr}_{E/k}(e(u \circ v)) = \beta(ue, v) \tag{2.1}$$

Thus for all $u_1, u_2, v \in V$ we have

$$\begin{aligned}
\mathrm{Tr}_{E/k}(e((u_1 + u_2) \circ v)) &= \beta((u_1 + u_2)e, v) \\
&= \beta(u_1 e, v) + \beta(u_2 e, v) \\
&= \mathrm{Tr}_{E/k}(e(u_1 \circ v)) + \mathrm{Tr}_{E/k}(e(u_2 \circ v)) \\
&= \mathrm{Tr}_{E/k}(e(u_1 \circ v + u_2 \circ v))
\end{aligned}$$

whence

$$(u_1 + u_2) \circ v = u_1 \circ v + u_2 \circ v.$$

Furthermore,

$$\mathrm{Tr}_{E/k}(e_1 e_2(u \circ v)) = \beta(u e_1 e_2, v) = \mathrm{Tr}_{E/k}(e_1(u e_2 \circ v))$$

and therefore

$$u e_2 \circ v = (u \circ v) e_2.$$

In addition

$$\begin{aligned}
\mathrm{Tr}_{E/k}(e(\overline{u \circ v})) &= \mathrm{Tr}_{E/k}(\bar{e}(u \circ v)) \\
&= \beta(u\bar{e}, v) = \beta(u, ve) = \beta(ve, u) \\
&= \mathrm{Tr}_{E/k}(e(v \circ u))
\end{aligned}$$

and therefore $\overline{u \circ v} = v \circ u$, which completes the proof that $u \circ v$ is hermitian.

Taking $e = 1$ in (2.1) we have $\beta(u, v) = \mathrm{Tr}_{E/k}(u \circ v)$ and therefore $u \circ v$ is non-degenerate.

If $u \cdot v$ is another hermitian inner product on V such that $\beta(u, v) = \mathrm{Tr}_{E/k}(u \cdot v)$, then $\mathrm{Tr}_{E/k}(e(u \cdot v)) = \mathrm{Tr}_{E/k}(ue \cdot v) = \beta(ue, v) = \mathrm{Tr}_{E/k}(e(u \circ v))$ whence $u \cdot v = u \circ v$. □

Remark 2.2. Suppose that $m(t) \in k[t]$ is an irreducible $*$-symmetric polynomial of degree at least 2. It follows from Lemma 1.2 (iii) that the degree of m is even. Let H be a vector space over the field $E = k[t]/(m(t))$ and let $u \circ v$ be a non-degenerate hermitian form on H.

Then $\beta(u, v) = \mathrm{Tr}_{E/k}(u \circ v)$ is a non-degenerate symmetric bilinear form on the space H regarded as a k-space.

If $\theta = t + (m(t))$, then $m(\theta^{-1}) = 0$ and $\theta \mapsto \theta^{-1}$ extends to an automorphism of E. Multiplication by θ satisfies $\beta(u\theta, v\theta) = \beta(u, v)$ and hence θ defines an element of the orthogonal group of β. The minimal polynomial of θ is $m(t)$.

For all non-degenerate unitary spaces H the Witt index of β is maximal if and only if $\dim_E(H)$ is even. (See the proof of Theorem 3.8.)

3 Orthogonal decompositions

In this section we show that for $g \in \mathrm{GO}^{\varepsilon}(V)$ the direct sum decomposition (1.1)

$$V_g = \bigoplus_{f \in \Phi, i} k[t]/(f)^{\mu_i(f)}$$

can be converted to an orthogonal decomposition and the calculation of the conjugacy class of g can be reduced to studying the restriction of g to each component.

3.1 Primary components

Definition 3.1. For each irreducible polynomial $f(t)$, the f-primary component of (1.1) is

$$V_{(f)} = \bigoplus_i k[t]/(f)^{\mu_i(f)} = \{\, v \mid vf(g)^i = 0 \text{ for sufficiently large } i \,\}.$$

Lemma 3.2. *$V_{(f)}$ is orthogonal to $V_{(h)}$ unless $h(t) = f^*(t)$.*

Proof. (Milnor [12]) If $u \in V_{(f)}$ and $v \in V$, then for sufficiently large i

$$\beta(u, vf(g^{-1})^i) = \beta(uf(g)^i, v) = 0$$

and hence $V_{(f)}$ is orthogonal to $Vf^*(g)^i$.

If $f^*(t) \neq h(t)$, then by irreducibility there are polynomials $r(t)$ and $s(t)$ such that $1 = r(t)h(t)^i + s(t)f^*(t)$. It follows that for large i and for $v \in V_{(h)}$ we have $v = vs(g)f^*(g)$ and therefore the map

$$V_{(h)} \to V_{(h)} : v \mapsto vf(g^{-1})$$

is a bijection. Hence $V_{(f)}$ is orthogonal to $V_{(h)}$. □

Corollary 3.3. *$V = \perp_f \widetilde{V}_{(f)}$, where f ranges over all $*$-irreducible polynomials and where*

$$\widetilde{V}_{(f)} = \begin{cases} V_{(f)} & f = f^* \text{ is irreducible;} \\ V_{(h)} \oplus V_{(h^*)} & f = hh^* \text{ and } h \neq h^*. \end{cases}$$

Lemma 3.4. *If $f(t)$ irreducible but not $*$-symmetric, then $V_{(f)}$ and $V_{(f^*)}$ are totally isotropic and $V_{(f)} \oplus V_{(f^*)}$ is non-degenerate.*

Proof. Let $U = V_{(f)}$ and write $V = U \oplus W$, where W is the sum of the h-primary components with $h \neq f$. Then $U^* = W^\perp$ is a $k[t]$-submodule and $\dim U^* = \dim U$.

For all $u \in U$, $v \in U^*$ and $i \geq 1$ we have

$$\beta(uf(g)^i, v) = 0 \quad \text{if and only if} \quad \beta(u, vf^*(g)^i) = 0$$

and therefore $f(g)^i$ vanishes on U if and only if $f^*(g)^i$ vanishes on U^*. (This is a consequence of the equalities $W = W^{\perp\perp}$, $U^\perp \cap W^\perp = 0$ and $U \cap W = 0$.) It follows that $U^* = V_{(f^*)}$. But $V_{(f^*)} \subseteq W$ and hence U^* is totally isotropic. Reversing the rôles of U and U^* we see that U is also isotropic. It is now clear that $U \oplus U^*$ is non-degenerate. □

Given a matrix X, the PRIMARYRATIONALFORM(X) intrinsic returns its rational form C, a transformation matrix T and the sequence *p*FACT of primary invariant factors. The entries in *p*FACT are pairs $\langle f, e \rangle$, where f is an irreducible polynomial and e is an integer. If the polynomials are $f_1, f_2, \ldots, f_r$ and if the entries with polynomial f_i are $\langle f_i, e_{i1} \rangle$, $\langle f_i, e_{i2} \rangle$, $\ldots, \langle f_i, e_{is} \rangle$, then we rely on MAGMA to group all pairs with the same irreducible polynomials and to order them so that $e_{i1} \leq e_{i2} \leq \cdots \leq e_{ir}$.

Assuming this is the case, the function *primaryParts* returns the list of $*$-irreducible polynomials, the corresponding list of partitions and a list of row indices giving the location of each primary component. This is almost all that is needed to construct the conjugacy class invariant for X. The complete invariant needs signs attached to the partitions associated with $t - 1$ and $t + 1$.

```
primaryParts := function(pFact)
   P := Parent(pFact[1][1]);
   pols := [P| ];
   parts := [];
   duals := [P| ];
   rows := [];
   j := 1;
   rownum := 0;
   for i := 1 to #pFact do
      f := pFact[i][1]; ndx := pFact[i][2];
      if f eq DualPolynomial(f) then
         if j eq 1 or pols[j–1] ne f then
            pols[j] := f;
            parts[j] := [];
            rows[j] := [];
            j +:= 1;
         end if;
         Append(~parts[j–1], ndx);
         r := j – 1;
      elif f notin duals then     // skip if in duals
         h := DualPolynomial(f);
         if IsEmpty(duals) or h ne duals[#duals] then
            Append(~duals, h);
            pols[j] := h*f;
            parts[j] := [];
            rows[j] := [];
            j +:= 1;
         end if;
         Append(~parts[j−1], ndx);
         r := j − 1;
      else
         h := DualPolynomial(f);
         r := Index(pols, f*h);
      end if;
      m := Degree(f)*ndx;
      rows[r] cat:= [rownum + i : i in [1..m]];
      rownum +:= m;
   end for;
   return pols, parts, rows;
end function;
```

As in Milnor [12] we divide the primary components $\widetilde{V}_{(f)}$, where $f(t)$ is $*$-irreducible, into three types:

Type 1. $f(t) = h(t)h^*(t)$ and $h(t) \neq h^*(t)$.

Type 2. $f(t) = f^*(t)$ is irreducible and the degree of $f(t)$ is even.

Type 3. $f(t) = f^*(t) = t \pm 1$.

By Corollary 3.3, we have an orthogonal splitting $V = \perp_f \widetilde{V}_{(f)}$. The subspaces $V_{(t-1)}$ and $V_{(t+1)}$ can be found using the matrix T from the primary rational form. Suppose, for example, that $t+1$ occurs in the decomposition and that the corresponding portion of the rational form occupies rows $a+1, a+2, \ldots, a+m$ of C. Since $TX = CT$ the rows $T[a+1]$, $T[a+2], \ldots, T[a+m]$ of T are a basis for $V_{(t+1)}$.

Type 1 companion matrices

For $\widetilde{V}_{(f)}$ of type 1, if we choose a basis $v_1, v_2, \ldots, v_r$ for $V_{(h)}$ and the basis $w_1, w_2, \ldots, w_r$ for $V_{(h^*)}$ such that $\beta(v_i, w_{r-j+1}) = \delta_{ij}$, the matrices of β and g restricted to $\widetilde{V}_{(f)}$ are

$$\begin{pmatrix} 0 & \Lambda \\ \Lambda & 0 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} A & 0 \\ 0 & \Lambda A^{-\mathrm{tr}}\Lambda \end{pmatrix}.$$

The minimal polynomial of A is $h(t)^s$ for some s and the minimal polynomial of A^{-1} is $h^*(t)^s$. If $\mu(h) = (\mu_1, \mu_2, \ldots)$ is the partition determined by A (in the general linear group), the conjugacy class of $g\,|_{\widetilde{V}_{(f)}}$ is completely determined by the pair $\langle f, \mu(h)\rangle$, where $f(t) = h(t)h^*(t)$. Note that $\Lambda^{-1} = \Lambda^{\mathrm{tr}} = \Lambda$. The Witt type of $\widetilde{V}_{(f)}$ is $\langle 0\rangle$.

If $h(t) = h_1(t)^e$, where $h_1(t)$ is an irreducible polynomial of degree d, which may or may not be $*$-symmetric, the following code returns a matrix in $\mathrm{GO}^+(2de, q)$ whose primary invariants are $[\langle h, e\rangle, \langle h^*, e\rangle]$.

```
type1Companion := function(h)
   d := Degree(h);
   A := CompanionMatrix(h);
```

```
    Λ := ZeroMatrix(BaseRing(h), d, d);
    for i := 1 to d do Λ[i, d−i+1] := 1; end for;
    return DiagonalJoin(A, Λ∗Transpose(A^−1)∗Λ);
end function;
```

Orthogonal splitting of a primary component

If $V_{(f)}$ is a primary component of type 2 or 3, then $V_{(f)}$ is an orthogonal summand of V_g. Throughout this section we suppose that V_g has a single primary component $V_{(f)}$; that is, the minimal polynomial of g is a power of the $*$-symmetric irreducible polynomial $f(t)$.

Lemma 3.5. *If f is irreducible and $*$-symmetric, $V_{(f)}$ splits as an orthogonal sum $V_{(f)} = V^1 \perp V^2 \perp \cdots \perp V^r$, where each V^i is annihilated by $f(g)^i$ and is free as a module over $k[t]/(f^i)$.*

Proof. (Milnor [12]) From the Jordan decomposition we have $V_{(f)} = W_1 \oplus W_2 \oplus \cdots \oplus W_r$ with W_i free as a $k[t]/(f^i)$-module but where the decomposition may not be orthogonal. Suppose that $W_r \cap W_r^\perp \neq 0$. Since $W_r \cap W_r^\perp$ is g-invariant we may choose $u \in W_r \cap W_r^\perp$ such that $u \neq 0$ and $uf(g) = 0$. But then $u = vf(g)^{r-1}$ for some $v \in W_r$. For $i < r$ and $w \in W_i$ we have

$$\beta(u, w) = \beta(vf(g)^{r-1}, w) = \beta(v, wf(g^{-1})^{r-1}) = 0$$

because $f = f^*$ and $i < r$. Thus u is in the radical of β, which is a contradiction. It follows that $V_{(f)} = W_r^\perp \perp W_r$ and the proof is complete by induction. □

Definition 3.6. The $k[t]$-modules V^i are the *homocyclic* components of $V_{(f)}$.

3.2 Primary components of type 2

Lemma 3.7. *Suppose that $V_{(f)}$ is a primary component of type 2 and define $s(t) = f(t)t^{-d}$, where the degree of $f(t)$ is $2d$. Then $s(g)$ is self-adjoint; that is, for all $u, v \in V_{(f)}$ we have*

$$\beta(us(g), v) = \beta(u, vs(g)).$$

Proof. For $u, v \in V_{(f)}$ it follows from equation (1.3), the assumption $f(t) = f^*(t)$ and the fact that $f(0) = 1$, that

$$\begin{aligned}\beta(us(g), v) &= \beta(uf(g)g^{-d}, v) = \beta(u, vg^d f(g^{-1})) \\ &= \beta(u, vg^{-d} f(g)) = \beta(u, vs(g)).\end{aligned}$$ □

Theorem 3.8 (Wall [17, p. 23], Milnor [12], Britnell [3, p. 91])**.** *Suppose that g acts cyclically on V with minimal polynomial $f(t)^e$, where $f(t)$ is $*$-symmetric and irreducible of degree $2d$. If $i = \lfloor (e+1)/2 \rfloor$, then $Vs(g)^i$ is a totally isotropic subspace of V of dimension $2d(e-i)$. Furthermore the Witt index of V is maximal if and only if e is even.*

Proof. For all $u, w \in V$ we have $\beta(us(g)^i, ws(g)^i) = \beta(u, ws(g)^{2i}) = 0$ and therefore $W = Vs(g)^i$ is totally isotropic.

If v is a generator of V, we shall show that the vectors $vs(g)^i, vs(g)^i g, \ldots, vs(g)^i g^{2d(e-i)-1}$ are linearly independent. To this end, suppose there is a polynomial $h(t)$ of degree at most $2d(e-i) - 1$ such that $vs(g)^i h(g) = 0$. Then $f(t)^i h(t)$ vanishes on V, hence $f(t)^e$ divides

$f(t)^i h(t)$ and so $f(t)^{e-i}$ divides $h(t)$. On comparing degrees we see that $h(t) = 0$ and thus the vectors are linearly independent. Thus $\dim W \geq 2d(e-i)$.

If $e = 2i$, then $\dim V = 4di$ and $\dim W \leq 2di$. It follows that $\dim W = \frac{1}{2}\dim V$ and therefore the Witt index of V is maximal.

For the remainder of the proof we may suppose that $e = 2i-1$. For all $u, w \in V$ we have $\beta(us(g)^i, ws(g)^{i-1}) = \beta(u, vs(g)^{2i-1}) = 0$ and consequently $W \subseteq Vs(g)^{i-1} \subseteq W^\perp$. Repeating an argument used above we see that the vectors $vs(g)^{i-1}, vs(g)^{i-1}g, \ldots, vs(g)^{i-1}g^{2di-1}$ are linearly independent and therefore $\dim Vs(g)^{i-1} \geq 2di$. We have $\dim V = 2d(2i-1)$ and therefore $\dim W^\perp \leq 2di$, whence equality holds. Thus $\dim Vs(g)^{i-1} = \dim W^\perp = 2di$ and so $Vs(g)^{i-1} = W^\perp$, $\dim W = 2d(i-1)$ and $\dim W^\perp/W = 2d$.

It now follows that the bilinear form β induces a well-defined, non-degenerate, symmetric form on $W^\perp/W$. Furthermore V and $W^\perp/W$ have the same Witt index. Thus from now on we may suppose that $e = 1$.

From section 2 the field $E = k[t]/f(t)$ has an automorphism $a \mapsto \bar{a}$, with fixed field E_0, such that $\beta(ua, w) = \beta(u, w\bar{a})$ and $\beta(va_1, va_2) = \mathrm{Tr}_{E/k}(a_1\bar{a}_2)$.

The kernel of the norm map $\mathrm{N}_{E/E_0} : E^\times \to E_0^\times$ has size $q^d + 1$ and the kernel of the trace $\mathrm{Tr}_{E/k}$ restricted to E_0 has size q^{d-1}. Thus there are $(q^d+1)(q^{d-1}-1)$ singular vectors in V. It follows from [16, Theorem 11.5] that the Witt index of V is $d-1$. □

Corollary 3.9. *Suppose that V^e is a homocyclic component of type 2.*

(a) *If $e = 2i$, then $V^{2i}s(g)^i$ is a maximal totally isotropic subspace.*

(b) *If e is odd, then V^e has maximal Witt index if and only if it is the sum of an even number of copies of $k[t]/(f^e)$.*

Theorem 3.10 (Milnor [12]). *Suppose that $V_{(f)} = V^1 \perp V^2 \perp \cdots \perp V^r$ is a primary component of type 2 where V^i is free as a $k[t]/(f(t)^i)$-module and $E = k[t]/(f(t))$. Then for all i the E-space $H^i = V^i/V^i f(g)$ carries a unique hermitian form $(u) \circ (v)$ such that*

$$\beta(us(g)^{i-1}, v) = \mathrm{Tr}_{E/k}((u) \circ (v)).$$

Proof. If $V(i) = \{v \in V \mid vf(g)^i = 0\}$, then $V^i/V^i f(g) \cong V(i)/(V(i-1) + V(i+1)f(g))$ and so the E-space H^i depends only on V and g. Furthermore, since $f(t)$ is the minimal polynomial of the induced action of g, the results of section 2 apply to H^i.

From the previous lemma, for $u, v \in V(i)$ the bilinear form $\beta(us(g)^{i-1}, v)$ is symmetric and depends only on the images (u) and (v) of u and v modulo $V(i-1) + V(i+1)f(g)$. Thus the result follows from Lemma 2.1. □

Milnor [12] shows that the sequence of unitary spaces H^1, H^2, … forms a complete invariant for the pair $V_{(f)}, g \mid V_{(f)}$.

Type 2 companion matrices, even exponent

A primary component of type 2 corresponds to an irreducible $*$-symmetric polynomial $f(t)$ of degree $2d$. The space $V_{(f)}$ is an orthogonal sum of components V^e, where V^e is a free $k[t]/(f^e)$-module. We first deal with the case in which $e = 2r$ is even. Then multiplication by $t + (f^e)$ on $k[t]/(f^e)$ is a linear transformation in $\mathrm{GO}^+(2de, q)$. To construct the matrix C of this linear transformation we begin with the companion matrix A of $f(t)^r$. Thus

$$A = \begin{pmatrix} 0 & 1 & & & & \\ & 0 & 1 & & & \\ & & 0 & \ddots & & \\ & & & \ddots & 1 & \\ & & & & 0 & 1 \\ & & & & & 0 \\ -1 & -a_1 & -a_2 & \cdots & -a_2 & -a_1 \end{pmatrix} \quad \text{and} \quad C = \begin{pmatrix} A & B \\ 0 & \Lambda_{de} A^{-\mathrm{tr}} \Lambda_{de} \end{pmatrix}$$

preserves the form Λ_{2de} if and only if $B = SA^{-\mathrm{tr}}\Lambda_{de}$ where S is antisymmetric. We choose $S = (s_{ij})$ so that every entry is 0 except that $s_{de-1,de} = -s_{de,de-1} = 1$. Then $f(C)^r = \begin{pmatrix} 0 & B' \\ 0 & 0 \end{pmatrix}$ where $B'_{1,1} = -1$. Therefore the minimal polynomial of C is $f(t)^e$. [Argue by induction on i (for $1 \le i \le de$) that the first non-zero entry in column $de + 1$ of A^i is -1 in row $de - i + 1$.]

```
type2CompanionEven := function(f, e);
  assert IsEven(e);
  r := e div 2;
  n := r*Degree(f);
  F := BaseRing(f);
  A := CompanionMatrix(f^r);
  L := StandardSymmetricForm(n, F);
  S := ZeroMatrix(F, n, n);
  S[n-1, n] := 1; S[n, n-1] := -1;
  AL := Transpose(A^-1)*L;
  return BlockMatrix(2, 2, [A, S*AL, 0, L*AL]);
end function;
```

Type 2 companion matrices, odd exponent

Now suppose that e is odd. The degree of $f(t)^e$ is $2de$ and

$$f(t)^e = 1 + a_1 t + a_2 t^2 + \cdots + a_r t^r + a_{r+1} t^{r+1} + \\ + (a_r + a_{r-1} t + \cdots + a_1 t^{r-1} + t^r) t^{r+2} \tag{3.1}$$

where $r = de - 1$.

If $e = d = 1$, then $a_1^2 - 4$ is not a square and hence there exists $b \in k$ such that $b^2 = \delta(a_1^2 - 4)$. The matrix $\frac{1}{2}\begin{pmatrix} -a_1 & b/\delta \\ b & -a_1 \end{pmatrix}$ preserves the form $\begin{pmatrix} 1 & 0 \\ 0 & -\delta \end{pmatrix}$ and its minimal polynomial is $x^2 + a_1 x + 1$.

If $de > 1$ the following matrix is a variant of one that appears in [13].

$$C_f = \left(\begin{array}{cccc|cc|cccc}
0 & 1 & & & & & & & & \\
 & 0 & \ddots & & & & & & & \\
 & & \ddots & 1 & & & & & & \\
 & & & 0 & & & & & & 1/b_0 \\
\hline
 & & & & 1 & 0 & & & & (-1)^r/2b_0 \\
 & & & & 0 & -1 & & & & -(-1)^r\gamma/2b_0 \\
\hline
b_0 & b_1 & \cdots & b_{r-1} & -(-1)^r & (-1)^r & 0 & \cdots & 0 & -\tau \\
 & & & & & & 1 & \ddots & & -b_{r-1}/b_0 \\
 & & & & & & & \ddots & 0 & \vdots \\
 & & & & & & & & 1 & -b_1/b_0
\end{array}\right).$$

In this matrix $\gamma = (-1)^{r+1} f(-1)^e / f(1)^e$, $\tau = a_r + a_{r-2} + \cdots + a_\epsilon$ and $\epsilon \in \{0, 1\}$ satisfies $\epsilon \equiv r \pmod 2$. The quantities $b_0, b_1, \ldots, b_{r-1}$ satisfy $b_0 = f(1)^{-1}$, $b_1 = a_1 b_0$ and $b_i = a_i b_0 + b_{i-2}$ for $i \geq 2$.

When $d = 2$ and $e = 1$ the matrix reduces to

$$C_f = \left(\begin{array}{c|cc|c}
0 & & & f(1) \\
\hline
 & 1 & 0 & -f(1)/2 \\
 & 0 & -1 & \gamma f(1)/2 \\
\hline
f(1)^{-1} & 1 & -1 & -\tau
\end{array}\right).$$

A direct calculation shows that C_f preserves the symmetric bilinear form represented by

$$J = \begin{pmatrix} 0 & 0 & 0 & \Lambda_{de} \\ 0 & 1/2 & 0 & 0 \\ 0 & 0 & -\gamma/2 & 0 \\ \Lambda_{de} & 0 & 0 & 0 \end{pmatrix},$$

It follows from Theorem 3.8 that the Witt index of J is $de - 1$.

Lemma 3.11. *Suppose that $f(t)$ is a monic irreducible $*$-symmetric polynomial of degree $2d$ and that the coefficients of $f(t)^e$, where e is odd, are given by* (3.1). *For $r = de - 1$ set*

$$\tau = a_r + a_{r-2} + \cdots + a_\epsilon \quad \textit{and}$$
$$\nu = a_{r+1} + 2(a_{r-1} + a_{r-3} + \cdots + a_{1-\epsilon})$$

where $\epsilon \in \{0, 1\}$ satisfies $\epsilon \equiv r \pmod 2$. Then $\tau \neq 0$, $\nu + 2\tau \neq 0$ and $\gamma = (\nu - 2\tau)/(\nu + 2\tau)$ is not a square in k. If γ_0 is a square root of γ in a quadratic extension K of k, then $c = \frac{1}{2}(1 + \gamma_0)$ satisfies $\tau \operatorname{Tr}_{K/k}(c^2) = \nu \operatorname{N}_{K/k}(c)$.

Proof. From Theorem 3.8 the Witt index of J is $de - 1$ and therefore γ is not a square in k.

Next observe that $\nu - 2\tau = (-1)^{r+1} f(-1)^e$ and that $\nu + 2\tau = f(1)^e \neq 0$. Let $\varphi \mapsto \bar{\varphi}$ be the automorphism of order 2 of K. A square root γ_0 of γ does not belong to k and therefore $\bar{\gamma}_0 = -\gamma_0$. Thus $c + \bar{c} = 1$, $c\bar{c} = \frac{1}{4}(1 - \gamma) = \tau/(\nu + 2\tau)$, whence $\tau \neq 0$. Furthermore, we have $c^2 + \bar{c}^2 = \nu/(\nu + 2\tau)$ and thus $\tau \operatorname{Tr}_{K/k}(c^2) = \nu \operatorname{N}_{K/k}(c)$. □

The following function is a modified and heavily optimised version of the function CMatO from `orthogonal.m`.

```
type2CompanionOdd := function(f, e);
  assert IsOdd(e);
  k := BaseRing(f);
  f := f^e;
  n := Degree(f);
  δ := Nonsquare(k);
  if n eq 2 then
    a_1 := Coefficient(f, 1);
    d := Sqrt(δ*(a_1^2 − 4));
    return Matrix(2, 2, [k| −a_1/2, d/(2*δ), d/2, −a_1/2]);
  end if;

  r := n div 2 − 1;
  as := Coefficients(f);
  τ := &+[ as[r−2*i+1] : i in [0. . r div 2] ];
  ν := as[r+2] + 2 * &+[ as[r+2−2*i] : i in [1. . (r+1) div 2] ];
  b_0 := (ν+2*τ)^−1;
  γ := (ν − 2*τ)*b_0;
  bs := [ (i gt 2) select as[i]*b_0 + Self(i−2) else
          (i eq 2) select as[2]*b_0 else b_0 : i in [1. . r]];
```

Construct the matrix C.

```
  C := ZeroMatrix(k, n, n);
  for i := 1 to r−1 do C[i, i+1] := 1; end for;
  C[r, n] := 1/b_0;
  C[r+1, r+1] := 1;
  C[r+1, n] := (−1)^r/(2*b_0);
  C[r+2, r+2] := −1;
  C[r+2, n] := −(−1)^r*γ/(2*b_0);
  for i := 1 to r do C[r+3, i] := bs[i]; end for;
  C[r+3, r+1] := −(−1)^r;
  C[r+3, r+2] := (−1)^r;
  C[r+3, n] := −τ;
  for i := 2 to r do
    C[r+2+i, r+1+i] := 1;
    C[r+2+i, n] := −bs[r+2−i]/b_0;
  end for;
```

The matrix C preserves the symmetric form whose central 2×2 block is $\begin{pmatrix} 1/2 & 0 \\ 0 & -\gamma/2 \end{pmatrix}$.
We return a conjugate that preserves the form with central block $\begin{pmatrix} 1 & 0 \\ 0 & -\delta \end{pmatrix}$.

```
  assert exists(a, b){<x, y> : x, y in k | x ne 0 and y ne 0 and x*x − y*y*γ eq 2};
  c := Sqrt(δ/γ);
  T := IdentityMatrix(k, n);
```

```
    T[r+1, r+1] := a; T[r+1, r+2] := b;
    T[r+2, r+1] := b*c*γ; T[r+2, r+2] := a*c;
    return T*C*T^-1, C;
end function;
```

3.3 Primary components of type 3

Suppose that $f(t) = t \pm 1$ and let $V^1 \perp V^2 \perp \cdots \perp V^r$ be an orthogonal decomposition of $V_{(f)}$ as in Lemma 3.5. The corresponding partition is a sequence of pairs $\langle i, m_i \rangle$, where $V^i \simeq m_i k[t]/(f(t)^i)$. Note that we may have $m_i = 0$ for some i.

Lemma 3.12. *If $\Delta = g - g^{-1}$, then $\beta(u\Delta, v) = -\beta(u, v\Delta)$.* □

Define $\overline{V}_i = V^i / V^i f(g)$. Then $\dim \overline{V}_i = m_i$. For $v \in V^i$, let (v) denote its image in $\overline{V}_i$ and for $(u), (v) \in \overline{V}_i$ define

$$(u) \circ (v) = \beta(u\Delta^{i-1}, v).$$

Theorem 3.13 (Milnor [12]). *The bilinear form $(u) \circ (v)$ is well-defined and non-degenerate. If i is odd it is symmetric, whereas if i is even it is alternating and hence m_i is even. Furthermore, the sequence consisting of the isomorphism classes of these symplectic and quadratic spaces $\overline{V}_i$ forms a complete invariant for the isomorphism class of $V_{(f)}$, $g \mid V_{(f)}$.*

Type 3, symplectic type

If e is even, a matrix representing the action of g on V^e can be obtained by repeated application of *type1Companion*. Alternatively we may use the following code.

The 'standard' Jordan block of size n for the scalar a is the $n \times n$ matrix with a along the diagonal, 1s on the upper diagonal and 0 elsewhere. Its primary invariant is $(t - a)^n$.

```
stdJordanBlock := function(n, a)
    D := ScalarMatrix(n, a);
    for i := 1 to n–1 do D[i, i+1] := 1; end for;
    return D;
end function;
```

Here is the code to produce a companion matrix for < $t + a_0$, [<e ,2>] >, where e is even and $a_0 = \pm 1$. This is a variant of *type1Companion* because in this case $\Lambda B^{-\mathrm{tr}} \Lambda = B^{-1}$.

```
type3CompanionS := func< a_0, e | DiagonalJoin(B, B^-1) where
        B is stdJordanBlock(e, –a_0) >;
```

The Witt type of this matrix is $\langle 0 \rangle$.

Type 3, orthogonal type

Assume that the characteristic of k is odd. In addition, for the remainder of this section suppose that e is odd and that the dimension of the quadratic space $\overline{V_e}$ is m. We may take the quadratic form to be $Q((v)) = \frac{1}{2}(v)\circ(v)$ and write $\overline{V_e}$ as an orthogonal sum of 1-dimensional subspaces.

In this case there are two conjugacy classes of elements in the orthogonal group with the same minimal polynomial $(t+a_0)^e$ and multiplicity m. As indicated above, in order to distinguish between these classes, one class will be represented by $\langle e, m\rangle$ and the other by $\langle -e, m\rangle$.

Definition 3.14.

(i) If m is even, the sign is $+1$ if the Witt type of $\overline{V_e}$ is $\langle 0\rangle$ and -1 otherwise.

(ii) If m is odd, there are two isomorphism classes of quadratic spaces $\overline{V_e}$, which have the same group of isometries but are distinguished by the discriminant of the symmetric form $(u)\circ(v)$; equivalently by the Witt type of $\overline{V_e}$. If the Witt type is $\langle 1\rangle$, the *sign* is $+1$ and -1 otherwise.

The discriminant of a hyperbolic plane is $-1 \pmod{k^2}$ and the discriminant of a 2-dimensional quadratic space with no isotropic vectors is $-\delta \pmod{k^2}$, where δ is a non-square in k.

Consequently, if m is even and $\overline{V_e}$ has maximal Witt index, the discriminant is $(-1)^{m/2} \pmod{k^2}$ whereas if the Witt index is not maximal, the discriminant is $(-1)^{m/2}\delta \pmod{k^2}$.

For example, if $m \equiv 0 \pmod 4$ or if $q \equiv 1 \pmod 4$ the discriminant is a square if and only if the Witt index is maximal.

The following function returns a representative for $< t + a_0, [<e, 1>] >$, where $|e| = 2c+1$ and $a_0 = \pm 1$. The return value is $g = \begin{pmatrix} -a_0 B & u & \frac{1}{2}a_0 bS \\ 0 & -a_0 & -b(\Lambda B^{-1}u)^{\mathrm{tr}} \\ 0 & 0 & -a_0 B^{-1}\end{pmatrix}$, where B is a standard $c \times c$ Jordan block all of whose non-zero entries are 1, u is the transpose of $(0, 0, \ldots, 0, 1)$ and b is 1 if $e > 0$ and δ otherwise. All entries in S are 0 except for its last row, which alternates between 1 and -1.

```
type3CompanionO := function($a_0$, $e$)
  assert IsOdd($e$);
  $c$ := (Abs($e$) − 1) div 2;
  $F$ := Parent( $a_0$ );
  if $c$ eq 0 then return Matrix($F$,1,1, [$-a_0$]); end if;
  $b$ := $F$ ! 1;
  $B$ := stdJordanBlock($c, b$);
  $X$ := $-a_0*$DiagonalJoin(<$B$, Matrix(1,1,[$b$]), $B^{-1}$>);
  $a$ := $a_0/2$;
  if ($e$ lt 0) then
    $b$ := Nonsquare($F$);
    $a$ *:= $b$;
  end if;
  $X[c, c+1]$ := 1;
```

```
    for $i$ := 1 to $c$ do
      $X[c, c+1+i]$ := IsOdd($i$) select $a$ else $-a$;
      $X[c+1, c+1+i]$ := IsOdd($i$) select $-b$ else $b$;
    end for;
    return $X$;
  end function;
```

The Witt type of the matrix g returned by this function is $\langle\, 1 \,\rangle$ if $e > 0$ and it is $\langle\, \delta \,\rangle$ if $e < 0$. If $\Delta = g - g^{-1}$ the only non-zero entry in $\Delta^{|e|-1}$ is at the top right corner.

4 Orthogonal direct sums

If matrices A_1 and A_2 preserve symmetric bilinear forms with matrices J_1 and J_2, their diagonal join preserves the diagonal join of J_1 and J_2. But, in general, this diagonal join will not be of 'standard' shape. Therefore we need functions to convert diagonal joins to the required shape, taking account the Witt types of the component forms.

The simplest case is when the Witt type of J_1 is $\langle 0 \rangle$ so that $J_1 = \begin{pmatrix} 0 & \Lambda_m \\ \Lambda_m & 0 \end{pmatrix}$ for some m, for then we may use the following 'central' join. If A_1 is a $2m \times 2m$ matrix and A_2 is an $n \times n$ matrix, we write A_1 as the block matrix

$$A_1 = \begin{pmatrix} P & Q \\ R & S \end{pmatrix}$$

and then the 'central join'

$$A_1 \circ A_2 = \begin{pmatrix} P & 0 & Q \\ 0 & A_2 & 0 \\ R & 0 & S \end{pmatrix}$$

belongs to the orthogonal group with the same Witt type as J_2 because

$$X^{-1} \begin{pmatrix} A_1 & 0 \\ 0 & A_2 \end{pmatrix} X = A_1 \circ A_2$$

where

$$X = \begin{pmatrix} I_m & 0 & 0 \\ 0 & 0 & I_m \\ 0 & I_n & 0 \end{pmatrix} \text{ so that } X^{-1} = X^{\mathrm{tr}} \text{ and } X^{\mathrm{tr}} \begin{pmatrix} J_1 & 0 \\ 0 & J_2 \end{pmatrix} X = \begin{pmatrix} 0 & 0 & \Lambda_m \\ 0 & J_2 & 0 \\ \Lambda_m & 0 & 0 \end{pmatrix}.$$

```
  centralJoin := function( $A$, $B$ )
    $d$ := NRows($A$);
    if $d$ eq 0 then return $B$; end if;
    $e$ := NRows($B$);
    if $e$ eq 0 then return $A$; end if;
    assert IsEven($d$);
    $m$ := $d$ div 2;
    $X$ := ZeroMatrix(BaseRing($A$), $d+e$, $d+e$);
    InsertBlock(~$X$, Submatrix($A$, 1, 1, $m$, $m$), 1, 1);
    InsertBlock(~$X$, Submatrix($A$, 1, $m$+1, $m$, $m$), 1, $m$+$e$+1);
    InsertBlock(~$X$, Submatrix($A$, $m$+1, 1, $m$, $m$), $m$+$e$+1, 1);
    InsertBlock(~$X$, Submatrix($A$, $m$+1, $m$+1, $m$, $m$), $m$+$e$+1, $m$+$e$+1);
    InsertBlock(~$X$, $B$, $m$+1, $m$+1);
    return $X$;
  end function;
```

The next step is to implement addition in the Witt ring $W(k)$. Recall that the *sign* of a quadratic space is $+1$ if its Witt type is $\langle 0\rangle$ or $\langle 1\rangle$ and that its sign is -1 if its Witt type is $\langle \delta\rangle$ or $\langle 1, -\delta\rangle$, where δ is a fixed non-square in k. Thus, as introduced in section 1.1, we may encode the Witt type as an integer $d \in \{-1, 0, 1, 2\}$, where $|d|$ is the dimension of the anisotropic kernel and, if d is odd, the sign of d is the sign of the Witt type.

```
  wittSum := function($F, d_1, d_2$)
    if $d_1$ eq 0 then $d$ := $d_2$;
    elif $d_2$ eq 0 then $d$ := $d_1$;
    elif #$F$ mod 4 eq 1 then
      $d$ := case< [$d_1, d_2$] |
        [1, 1]: 0, [−1, −1]: 0, [1, −1]: 2, [−1, 1]: 2,
        [1, 2]: −1, [2, 1]: −1, [−1, 2]: 1, [2, −1]: 1,
        [2, 2]: 0, default: "error">;
    else
      $d$ := case< [$d_1, d_2$] |
        [1, 1]: 2, [−1, −1]: 2, [1, −1]: 0, [−1, 1]: 0,
        [1, 2]: −1, [2, 1]: −1, [−1, 2]: 1, [2, −1]: 1,
        [2, 2]: 0, default: "error">;
    end if;
    return $d$;
  end function;
```

If X is the matrix returned by the following function, and if C is the diagonal join of matrices representing the Witt types d_1 and d_2 in the 'almost anti-diagonal shape' described in section 1, then XCX^{tr} represents the Witt sum in 'almost anti-diagonal shape'.

```
  wittTransform := function($F, d_1, d_2$)
    if $d_1$ eq 0 then return IdentityMatrix($F$, Abs($d_2$)); end if;
    if $d_2$ eq 0 then return IdentityMatrix($F$, Abs($d_1$)); end if;
    if #$F$ mod 4 eq 1 then
      $i$ := Sqrt(−$F$ ! 1);
      $\delta$ := Nonsquare($F$);
      $X$ := case< [$d_1, d_2$] |
        [1, 1]: Matrix($F$, 2, 2, [$F$| 1, $i$, 1/2, −$i$/2]),
        [−1, −1]: Matrix($F$, 2, 2, [$F$| 1, $i$, 1/(2∗$\delta$), −$i$/(2∗$\delta$)]),
        [1, −1]: Matrix($F$, 2, 2, [$F$| 1, 0, 0, $i$]),
        [−1, 1]: Matrix($F$, 2, 2, [$F$| 0, 1, $i$, 0]),
        [1, 2]: Matrix($F$, 3, 3, [$F$| 1, $i$, 0,  0, 0, $i$,  1/2, −$i$/2, 0]),
        [2, 1]: Matrix($F$, 3, 3, [$F$| 1, 0, $i$,  0, $i$, 0,  1/2, 0, −$i$/2]),
        [−1, 2]: Matrix($F$, 3, 3, [$F$|1, 0, 1,  0, 1, 0,  1/(2∗$\delta$), 0, −1/(2∗$\delta$)]),
        [2, −1]: Matrix($F$, 3, 3, [$F$| 0, 1, 1,  1, 0, 0,  0, −1/(2∗$\delta$), 1/(2∗$\delta$)]),
      [2,2]: Matrix($F$,4,4,[$F$| 1,0,$i$,0,  0,1,0,$i$,  0,$-1/(2*\delta)$,0,$i/(2*\delta)$,  1/2,0,$-i/2$,0]),
      default: “error”>;
  else
    assert exists($x,y$){ <$x,y$> : $x,y$ in $F$ | $x$ ne 0 and $y$ ne 0 and $x*x + y*y$ eq $-1$ };
    $X$ := case< [$d_1,d_2$] |
      [1,1]: IdentityMatrix($F$,2),
      [$-1,-1$]: Matrix($F$,2,2,[$F$| $x,y,\ y,-x$]),
      [$1,-1$]: Matrix($F$,2,2,[$F$| 1,1,1/2,$-1/2$]),
      [$-1,1$]: Matrix($F$,2,2,[$F$| 1,1,$-1/2$,1/2]),
      [1,2]: Matrix($F$,3,3,[$F$| $x,y$,1,  $y,-x$,0,  $-x/2,-y/2$,1/2]),
      [2,1]: Matrix($F$,3,3,[$F$| $x,y$,1,  $y,-x$,0,  $-x/2,-y/2$,1/2]),
      [$-1,2$]: Matrix($F$,3,3,[$F$|1,1,0,  0,0,1,  $-1/2$,1/2,0]),
      [$2,-1$]: Matrix($F$,3,3,[$F$| 1,0,1,  0,1,0,  1/2,0,$-1/2$]),
      [2,2]: Matrix($F$,4,4,[$F$| $x,y$,1,0,  $y,-x$,0,1,  $-y/2,x/2$,0,1/2,  $-x/2,-y/2$,1/2,0]),
      default: “error”>;
  end if;
  return $X$;
end function;
```

More generally we need a function to convert the diagonal join of matrices J_1 and J_2 to ‘almost anti-diagonal shape’: a matrix with every anti-diagonal entry equal to 1 except perhaps for a 1×1 or 2×2 central block, and 0 elsewhere. The following permutation will be used to carry out a change of basis from the diagonal join of two matrices in almost anti-diagonal shape to almost anti-diagonal shape. In this function n_1 and n_2 are the dimensions of the anisotropic kernels.

```
almostADPerm := function($m_1,m_2,n_1,n_2$)
  $X$ := [ $i$ : $i$ in [1..$m_1$] ] cat [ $2*m_1+n_1+i$ : $i$ in [1..$m_2$] ] cat
      [ $m_1+i$ : $i$ in [1..$n_1$] ] cat [ $2*m_1+n_1+m_2+i$ : $i$ in [1..$n_2$]] cat
      [ $2*m_1+n_1+m_2+n_2+i$ : $i$ in [1..$m_2$] ] cat [ $m_1+n_1+i$: $i$ in [1..$m_1$] ];
  return Sym($2*m_1+n_1+2*m_2+n_2$) ! $X$;
end function;
```

In the following function *mA* and *mB* are Witt indices and *dA* and *dB* are Witt codes.

```
oTransform := func< $F$, mA, mB, dA, dB | DiagonalJoin(<$Z$, wittTransform($F$, dA, dB), $Z$>)
    where $Z$ is IdentityMatrix($F$, mA+mB)>;
```

Suppose that J_1 and J_2 are the matrices (in ‘almost anti-diagonal shape’) of symmetric bilinear forms and that $g_i J_i g_i^{\mathrm{tr}} = J_i$ for $i = 1, 2$. Let J be the diagonal join of J_1 and J_2 and let g be the diagonal join of g_1 and g_2. If X is the matrix *oTransform*(F, m_1, m_2, d_1, d_2) where m_i is the Witt index of J_i and d_i is its Witt code, then $K = XJX^{\mathrm{tr}}$ is in almost anti-diagonal shape and XgX^{-1} preserves K.

For convenience we combine *centralJoin* and *oTransform* in the following function. If the parameter `Form` is `true`, the arguments *A* and *B* represent forms such as J_1 and J_2 as above. In this case the return value is the almost anti-diagonal sum J of J_1 and J_2. On the other hand, if `Form` is `false`, *A* and *B* represent matrices g_1 and g_2 which preserve J_1 and J_2. The return value is the almost anti-diagonal sum of g_1 and g_2, which preserves J.

```
orthogonalJoin := function(A, B, dA, dB : Form := false)
  rA := Nrows(A); rB := Nrows(B);
```

```
    if $rA$ eq 0 then return $B$; end if;
    if $rB$ eq 0 then return $A$; end if;
    if $dA$ eq 0 then return centralJoin($A$, $B$); end if;
    if $dB$ eq 0 then return centralJoin($B$, $A$); end if;
    $nA$ := Abs($dA$); $nB$ := Abs($dB$);
    $mA$ := ($rA - nA$) div 2; $mB$ := ($rB - nB$) div 2;
    $F$ := BaseRing($A$);
    $\pi$ := almostADPerm($mA, mB, nA, nB$);
    $C_0$ := DiagonalJoin($A$, $B$);
    $C$ := Matrix($F, n, n$, [$C_0[i^\pi, j^\pi]$ : $i, j$ in [1 . . $n$]]) where $n$ is $rA+rB$;
    $X$ := oTransform($F, mA, mB, dA, dB$);
    return (Form) select $X*C*$Transpose($X$) else $X*C*X^{-1}$;
end function;
```

5 Conjugacy classes in orthogonal groups (q odd)

If $f(t)$ is a polynomial of type 1, the companion matrices of the triples $\langle f, e, m \rangle$ all have Witt code equal to 0 and therefore they can be combined using the *centralJoin* function.

```
type1Matrix := function($f$, plist)
    factors := Factorisation($f$);
    $h$ := factors[1][1];
    assert $f$ eq $h*$factors[2][1];
    $X$ := ZeroMatrix( BaseRing($f$), 0, 0 );
    for $\mu$ in plist do
        $e$, $m$ := Explode($\mu$);
        for $i$ := 1 to $m$ do $X$ := centralJoin($X$, type1Companion($h^e$)); end for;
    end for;
    return $X$;
end function;
```

If f is a polynomial of type 2, the Witt code of a triple $\langle f, e, m \rangle$ is 0 if de is even and 2 if de is odd.

```
type2Matrix := function($f$, plist)
    $F$ := BaseRing($f$);
    $X$ := ZeroMatrix( $F$, 0, 0 );
    wCode := 0;
    for $\mu$ in plist do
        $e$, $m$ := Explode($\mu$);
        $d$ := 2*($e$ mod 2);
        t2c := IsEven($e$) select type2CompanionEven else type2CompanionOdd;
        for $i$ := 1 to $m$ do
            $X$ := orthogonalJoin($X$, t2c($f, e$), wCode, $d$);
            wCode := wittSum($F$, wCode, $d$);
        end for;
    end for;
    return $X$, wCode;
```

```
end function;

type3Matrix := function(f, plist)
   assert Degree(f) eq 1;
   assert isSignedPartition(plist);
   $a_0$ := Coefficient(f, 0);
   F := BaseRing(f);
   q := #F;
   X := ZeroMatrix( F, 0, 0 );
   wCode := 0;
   for $\mu$ in plist do
      e, m := Explode($\mu$);
```

If e is even, the Witt code of $\langle f, e, m \rangle$ is 0 and so the companion matrices may be combined using *centralJoin*.

```
         if IsEven( e ) then
            for i := 1 to (m div 2) do
               X := centralJoin( type3CompanionS($a_0$, e), X);
            end for;
         else
```

If e is odd, the Witt code of $\langle f, e, m \rangle$ depends on the sign of e as well as the parity of m and the sum of the codes depends on $q \pmod 4$.

```
            ae := Abs(e);
            r := (m − 1) div 2;
            mPlus := type3CompanionO($a_0$, ae);
            mMinus := type3CompanionO($a_0$, −ae);
```

The Witt code of Y should be 0.

```
            Y := (q mod 4 eq 1) select orthogonalJoin(mPlus, mPlus, 1, 1)
                   else orthogonalJoin(mPlus, mMinus, 1, −1);
            for i := 1 to r do X := centralJoin(Y, X); end for;
            if IsOdd(m) then
               if e gt 0 then
                  X := orthogonalJoin(X, mPlus, wCode, 1);
                  wCode := wittSum(F, wCode, 1);
               else
                  X := orthogonalJoin(X, mMinus, wCode, −1);
                  wCode := wittSum(F, wCode, −1);
               end if;
            else
               if e gt 0 then
                  X := centralJoin(Y, X);
               else
                  Y := (q mod 4 eq 3) select orthogonalJoin(mPlus, mPlus, 1, 1)
                         else orthogonalJoin(mPlus, mMinus, 1, −1);
                  X := orthogonalJoin(X, Y, wCode, 2);
                  wCode := wittSum(F, wCode, 2);
```

```
          end if;
        end if;
      end if;
    end for;
    return X, wCode;
  end function;
```

Class invariants and representatives

```
  intrinsic RepresentativeMatrixO( inv :: SetIndx[Tup] ) → GrpMatElt
  {A representative of the conjugacy class in the orthogonal
   group with invariant inv}
    F := BaseRing(Parent(inv[1][1]));
    X := ZeroMatrix(F, 0, 0);
    wCode := 0;
    for polpart in inv do
      f, plist := Explode(polpart);
      if (Degree(f) eq 1) then
        Y, d := type3Matrix(f, plist);
        X := orthogonalJoin(X, Y, wCode, d);
        wCode := wittSum(F, wCode, d);
      elif IsIrreducible(f) then
        Y, d := type2Matrix(f, plist);
        X := orthogonalJoin(X, Y, wCode, d);
        wCode := wittSum(F, wCode, d);
      else
        X := centralJoin(type1Matrix(f, plist), X);
      end if;
    end for;

    n := Nrows(X);
    if IsOdd(n) then
      q := #F;
      assert theSign(q, inv) eq 1;
      r := (n − 1) div 2;
      if (q mod 8) in [1, 7] then    // 2 is a square
        T := IdentityMatrix(F, n);
        T[r+1, r+1] := Sqrt(F ! 2);
      else
        $\delta$ := Nonsquare(F);
        a := Sqrt(2∗$\delta$);
        S := ScalarMatrix(r, $\delta$);
        T := DiagonalJoin(<S, Matrix(1, 1, [a]), IdentityMatrix(F, r)>);
      end if;
      X := $T^{-1}$∗X∗T;
    end if;
    return GL(n, F) ! X;

  end intrinsic;

  intrinsic ClassRepresentativesGO(d :: RngIntElt, q :: RngIntElt)
        -> SeqEnum, SeqEnum
  { Representatives for the conjugacy classes of the orthogonal
    group GO(d,q). The second return value is the sequence of
    class invariants. }
    require IsOdd(q) : "q must be odd";
    invar := ClassInvariantsGO(d, q);
    return [GO(d, q) | RepresentativeMatrixO(μ) : μ in invar], invar;
  end intrinsic;

  intrinsic ClassRepresentativesGOPlus(d :: RngIntElt, q :: RngIntElt)
        -> SeqEnum, SeqEnum
  { Representatives for the conjugacy classes of the orthogonal
    group GOPlus(d,q). The second return value is the sequence
    of class invariants. }
    require IsOdd(q) : "q must be odd";
    invar := ClassInvariantsGOPlus(d, q);
    return [GOPlus(d, q) | RepresentativeMatrixO(μ) : μ in invar], invar;
  end intrinsic;

  intrinsic ClassRepresentativesGOMinus(d :: RngIntElt, q :: RngIntElt)
        -> SeqEnum, SeqEnum
  { Representatives for the conjugacy classes of the orthogonal
    group GOMinus(d,q). The second return value is the sequence
    of class invariants. }
    require IsOdd(q) : "q must be odd";
    invar := ClassInvariantsGOMinus(d, q);
    return [GOMinus(d, q) | RepresentativeMatrixO(μ) : μ in invar], invar;
  end intrinsic;
```

6 The conjugacy class invariant of an orthogonal matrix

In previous sections we provided code to construct a representative of a conjugacy class invariant. The code in this section does the converse and computes the conjugacy class invariant of an orthogonal matrix.

Guided by Lemma 3.5 we shall define a function *homocyclicSplit* designed to be applied to a matrix g acting on a primary component $V_{(f)}$. But first we need the row indices for the homocyclic components of the rational canonical form of the matrix g. (We use this only when the polynomial is $t \pm 1$.)

```
getSubIndices := function(pFact)
    f := pFact[1][1];
    error if exists{ p : p in pFact | p[1] ne f },
      "the component is not homocyclic";
    d := Degree(f);
    ndx := 0;
    base := [];
    last := 0;
    rng := [];
    for j := 1 to #pFact do
      if j gt 1 and pFact[j][2] ne last then
        Append(~base, rng);
        rng := [];
      end if;
      last := pFact[j][2];
      n := last*d;
      rng cat:= [ndx+i : i in [1..n]];
      ndx +:= n;
    end for;
    Append(~base, rng);
    return base;
  end function;
```

We also need the restriction of a linear transformation (defined by a matrix M) to an invariant subspace; S is either the basis matrix for the subspace or a sequence of basis vectors. (There is no check that the subspace is invariant.)

```
  restriction := func< M, S | Solution(T, T*M) where T is Matrix(S) >;
```

In the following function W represents a primary component of g.

```
  homocyclicSplit := function(g, W)
    U := Universe([ W, sub<W|> ]);
    _, T, pFact := PrimaryRationalForm(g);
    baseNdx := getSubIndices(pFact);
    W_0 := sub< W | [T[i] : i in baseNdx[#baseNdx]] >;
    D := [U| W_0 ];
    while W ne W_0 do
      W0p := OrthogonalComplement(W, W_0);
      gp := restriction(g, BasisMatrix(W0p));
      _, T, pFact := PrimaryRationalForm(gp);
      baseNdx := getSubIndices(pFact);
      W_1 := sub< W | [T[i]*BasisMatrix(W0p) : i in baseNdx[#baseNdx]] >;
      Append(~D, W_1);
      W_0 := sub< W | W_0, W_1 >;
    end while;
    return Reverse(D);
  end function;
```

In the following function D is the subspace V^e obtained from *homocyclicSplit*, g is the matrix acting on the generic space of D, f is the polynomial $t+1$ or $t-1$ and μ is the pair $\langle e, m\rangle$. This function is only called if e is odd.

```
  attachSign := function(D, g, f, μ)
    F := BaseRing(g);
    e, m := Explode($\mu$);
    n := e*m*Degree(f);
    B := Matrix(F, n, n, [DotProduct(u, v) : u, v in Basis(D) ]);
    assert Determinant(B) ne 0;
    disc, _ := IsSquare(Determinant(B));
```

We attach a plus or minus sign to μ according to whether the Witt type of the quadratic space is $\langle 0 \rangle$ or $\langle 1 \rangle$. We can obtain this from the discriminant of B, taking into account that -1 is not a square in F when the size of F is congruent to 3 modulo 4.

```
    if ((#F mod 4 eq 3) and (n mod 4 in {2,3})) then disc := not disc; end if;
    return disc select $\mu$ else < −e, m>;
  end function;
```

Given an orthogonal matrix g, we find the invariant of its conjugacy class, following Wall [17] and Milnor [12]. First obtain the generalised Jordan decomposition and then treat the components whose minimal polynomials are powers of $t - 1$ or $t + 1$ separately.

```
  intrinsic ConjugacyInvariantO(g :: GrpMatElt : Minus := false) → SetIndx[Tup]
  { The conjugacy class invariant of the orthogonal matrix g }
    F := BaseRing(g);
    n := Nrows(g);
    Q := StandardQuadraticForm(n, F : Minus := Minus, Variant := "Revised");
    if IsOdd(n) then
      error if Minus, "Minus option not available in odd dimensions";
```

The revised variant of the standard quadratic form in odd dimensions creates a quadratic space with Witt type $\langle 1/2 \rangle$ and therefore we multiply by 2 to change its type to $\langle 1 \rangle$.

```
      Q := 2*Q;
    end if;
    V := QuadraticSpace(Q);
    require IsIsometry(V, g) :
      "matrix is not in the standard orthogonal group";
    _, T, pFact := PrimaryRationalForm(g);
    pols, parts, bases := primaryParts(pFact);
    inv := {@ @};
    for i := 1 to #pols do
      f := pols[i];
      plist := convert(parts[i]);
      if Degree(f) eq 1 then
        base := bases[i];
```

Extract the f-primary component W as a quadratic space with the g-action given by g_.

```
        g_ := restriction(g, [T[j] : j in base]);
        d := #base;
        Q := ZeroMatrix(F, d, d);
        for b := 1 to d−1 do
          Q[b, b] := QuadraticNorm(V ! T[base[b]]);
          for c := b+1 to d do
            Q[b, c] := DotProduct(V ! T[base[b]], V ! T[base[c]]);
          end for;
        end for;
        Q[d, d] := QuadraticNorm(V ! T[base[d]]);
        W := QuadraticSpace(Q);
        D := homocyclicSplit(g_, W);
        for j := 1 to #plist do
          if IsOdd(plist[j][1]) then
            plist[j] := attachSign(D[j], g_, f, plist[j]);
          end if;
        end for;
      end if;
      Include(~inv, <f, plist> );
    end for;
    return inv;
  end intrinsic;
```

Centraliser orders

The centraliser orders of the elements of orthogonal groups can be computed using Wall's functions $A(\varphi^{\mu})$ and $B(\varphi)$ from [17, §2.6]. Here f is a polynomial and $\langle e, m\rangle$ is a term from the partition list.

```
    A_fn := function(f, e, m, q)
      d := Degree(f);
      if IsIrreducible(f) then
        if d eq 1 then
          if IsEven(e) then val := OrderSp(m, q);
          else
            if IsOdd(m) then val := OrderGO(m, q);
            elif (e lt 0) then val := OrderGOMinus(m, q);
            else val := OrderGOPlus(m, q);
            end if;
          end if;
        else val := OrderGU(m, q^(d div 2));
        end if;
      else val := OrderGL(m, q^(d div 2));
      end if;
      return val;
    end function;

    κ := function(plist, d)
      val := 0;
      for μ in plist do
        e, m := Explode(μ);
        val +:= (Abs(e)−1)∗m^2;
        if d eq 1 and IsEven(e) then val −:= m; end if;
      end for;
      for i := 1 to #plist−1 do
      e := Abs(plist[i][1]);
      m := plist[i][2];
      for j := i+1 to #plist do val +:= 2*e*m*plist[j][2]; end for;
   end for;
   val *:= d;
   assert IsEven(val);
   return val div 2;
end function;
```

Here *pol_part* has the form $\langle f, [\ \ldots\ , \langle e, m \rangle, \ \ldots\ ] \rangle$.

```
B_fn := function(pol_part)
   f, plist := Explode(pol_part);
   q := #BaseRing(f);
   d := Degree(f);
   return $q^{\kappa(plist,d)}$ * &*[A_fn(f, $\mu$[1], $\mu$[2], q) : $\mu$ in plist];
end function;

intrinsic CentraliserOrderO( inv :: SetIndx[Tup] ) -> RngIntElt
{The order of the centraliser of any element in the orthogonal
 group whose conjugacy invariant is inv}
 return &*[ B_fn(pol_part) : pol_part in inv ];
end intrinsic;
```

7 The conjugacy classes of the general orthogonal groups

```
intrinsic ClassesGO(d :: RngIntElt, q :: RngIntElt) -> SeqEnum
{The conjugacy classes of GO(d,q)}
   ord := OrderGO(d, q);
   return Sort([car<Integers(), Integers(), GO(d, q)> |
      < Order(M), ord div CentraliserOrderO($\mu$), M > :
            $\mu$ in ClassInvariantsGO(d, q) | true
            where M is RepresentativeMatrixO($\mu$) ]);
end intrinsic;

intrinsic ClassesGOPlus(d :: RngIntElt, q :: RngIntElt) -> SeqEnum
{The conjugacy classes of GOPlus(d,q)}
   ord := OrderGOPlus(d, q);
   return Sort([car<Integers(), Integers(), GOPlus(d, q)> |
      < Order(M), ord div CentraliserOrderO($\mu$), M > :
            $\mu$ in ClassInvariantsGOPlus(d, q) | true
            where M is RepresentativeMatrixO($\mu$) ]);
end intrinsic;

intrinsic ClassesGOMinus(d :: RngIntElt, q :: RngIntElt) -> SeqEnum
{The conjugacy classes of GOMinus(d,q)}
   ord := OrderGOMinus(d, q);
   return Sort([car<Integers(), Integers(), GOMinus(d, q)> |
    < Order(M), ord div CentraliserOrderO($\mu$), M > :
        $\mu$ in ClassInvariantsGOMinus(d, q) | true
        where M is RepresentativeMatrixO($\mu$) ]);
end intrinsic;
```

8 Test code

```
Attach("common.m");
Attach("GOConjugacy.m");
```

The tests in this section compare the code developed in the current file with Scott Murray's implementation in `orthogonal.m`.

1. Scott's construction of all ϕ-irreducible polynomials.

```
AllIrreduciblePolynomialsL := function(F, i)
  P := PolynomialRing(F); X := P.1;
  return (i eq 1) select [ X−a : a in F | a ne 0 ]
    else SetSeq( AllIrreduciblePolynomials(F, i) );
end function;

HatSO := function( p )
    P := Parent( p );
    F := CoefficientRing( P );
    R<x> := RationalFunctionField( F );
    return P ! ( $x^{\text{Degree}(p)}$ ∗ Evaluate( R ! p, x + 1/x ) );
end function;

All1IrreduciblePolynomialsSO := function( F, d )
  P<X> := PolynomialRing(F);
  pols := {@@};
  if d eq 1 then
    pols := {@ X + a : a in {F|1, −1} @};
  elif IsEven(d) then
    allhalf := (d eq 2) select [ X + a : a in F | a ne 0] else
      AllIrreduciblePolynomialsL( F, d div 2 );
    if d eq 2 then
      Include( ~pols, HatSO(X) );
    end if;
    for f in allhalf do
      hat := HatSO(f);
      if IsIrreducible( hat ) then
        Include( ~pols, hat );
      end if;
    end for;
    for f in allhalf do
      dual := Dual( f );
      if f ne dual then
```

```
            Include( ~pols, f*dual );
         end if;
      end for;
   end if;
   return IndexedSetToSequence( pols );
end function;

AllTIrreduciblePolynomialsSO := function( F, d )
   P<X> := PolynomialRing(F);
   ts := [ t : t in F | t ne 0 ];
   pols := [ [] : t in ts ];
   if d eq 1 then
      for tidx in [1..#ts] do
         rts := Roots( $X^2 - ts[tidx]$ );
         pols[tidx] := [ X + rts[i][1] : i in [1..#rts] ];
      end for;
   elif IsEven(d) then
      for f in AllIrreduciblePolynomialsL( F, d ) do
         for i in [1..#ts] do
            if f eq Dual( f : t:=ts[i] ) then
               Append( ~pols[i], f );
            end if;
         end for;
      end for;
      for f in AllIrreduciblePolynomialsL( F, d div 2 ) do
         for i in [1..#ts] do
            dual := Dual( f : t:=ts[i] );
            if f ne dual and f*dual notin pols[i] then
            Append( ~pols[i], f*dual );
            end if;
         end for;
      end for;
   end if;
   return pols, ts;
end function;

$testgo_1$ := procedure(n, q)
   printf "1. *-irreducible polynomials for n=%o, q=%o\n", n, q;
   F := GF(q);
   time det := StarIrreduciblePolynomials(F, n);
   time shm, $\phi$ := AllTIrreduciblePolynomialsSO(F, n);
   detset := Seqset(det);
   shmset := Seqset(shm[1]);
   assert #det eq #shm[1];
   assert detset eq shmset;
   print "Passed\n";
end procedure;
```

```
testgo_1(4, 11);
```

2. Scott's construction of the signed partitions for the conformal orthogonal group. `partitions` is the list of all orthogonal signed partitions; `kill` says which to remove with one sign killed in the conformal group, i.e., partitions whose first signed part is negative; if `EvenChar` is true, only even parts with even multiplicity have signs.

```
SignedPartitionsO := function( d : EvenChar := false )
```

We create partitions starting with the smallest part size. When *part* = *i*, *oldpartitions* consists of a partitions with no part of size *i* or larger.

```
    oldpartitions := [ [] : i in [1. . d] ];
    oldkills := [ [] : i in [1. . d] ];
    for part in [1. . d] do
```

part = the size of the part we are currently adding

```
        partitions := oldpartitions; kills := oldkills;
        for n in [0. . d−1] do
```

currently adding to partitions of *n*

```
            dimleft := d−n;   // the amount of space left in our partition
            if part le dimleft then   // there is room
              if IsEven(part) then   // even parts have even multiplicity, no signs
                multleft := dimleft div (2∗part);   // there is room for multleft pairs of parts
                for i in [1. . multleft] do   // i = half the multiplicity of the new part
                  oldpartitionsn := ((n ne 0) select oldpartitions[n] else [[]]);
                  oldkillsn := ((n ne 0) select oldkills[n] else [false]);
                  for idx in [1..#oldpartitionsn] do   // for each old partition of n
                    partition := oldpartitionsn[idx];
                    kill := oldkillsn[idx];
                    Append( ~partitions[n+2∗part∗i], partition cat [<part, 2∗i>] );
                    Append( ~kills[n+2∗part∗i], kill );
                  end for;
                end for;
              else   // odd parts have signs
                multleft := dimleft div part;   // there is room for multleft parts
                for i in [1. . multleft] do   // i = the multiplicity of the new part
                  oldpartitionsn := ((n ne 0) select oldpartitions[n] else [[]]);
                  oldkillsn := ((n ne 0) select oldkills[n] else [false]);
                  for idx in [1..#oldpartitionsn] do   // for each old partition of n
                    partition := oldpartitionsn[idx];
                    kill := oldkillsn[idx];
                    Append( ~partitions[n+part∗i], partition cat [<part, i>] );
                    Append( ~kills[n+part∗i], kill );
                    Append( ~partitions[n+part∗i], partition cat [ <−part, i>] );
```

If the partition has not already been killed, and there is no earlier part which could have been killed . . .

```
                    if not kill and IsOdd(i) and
                    not exists{t : t in partition | IsOdd(t[1]) and IsOdd(t[2])}
                  then
                    kill := true;
                  end if;
                  Append( ~kills[n+part*i], kill );
                end for;   // idx
              end for;   // i
            end if;   // part even/odd
          end if;   // part le dimleft
        end for;   // n
        oldpartitions := partitions;
        oldkills := kills;
      end for;   // part
      return partitions, kills;
    end function;

    testgo_2 := procedure(d)
      printf "2. Signed partitions of %o\n", d;
      time det := signedPartitionsO(d);
      time shm, _ := SignedPartitionsO(d);
      assert #det eq #shm;
      for i := 1 to #det do
        assert Set(det[i]) eq Set(shm[i]);
      end for;
      print "Passed\n";
    end procedure;

    testgo_2(20);
```

3. Scott's construction of all partitions

```
    allPartitionsSHM := function( d )
      oldpartitions := [ [] : l in [1..d] ];
      for part in [1..d] do
        partitions := oldpartitions;
        for n in [0..d-1] do
          dimleft := d-n;
          if part le dimleft then
            multleft := dimleft div part;
            for i in [1..multleft] do
              for partition in ((n ne 0) select oldpartitions[n] else [[]]) do
                Append( ~partitions[n+part*i], partition cat [<part,i>] );
              end for;
            end for;
          end if;
        end for;
        oldpartitions := partitions;
      end for;
      return partitions;
    end function;
```

4. Scott's class parameters

Scott's class parameters are sequences of tuples $\langle f, \pi \rangle$, where f is a polynomial and π is a (signed) partition in multiplicity format. For example, $\pi = [\langle -1, 1 \rangle, \langle -3, 1 \rangle]$. In the present file these are referred to as class invariants and they are indexed sets rather than sequences of tuples.

```
  IsSignedPol := func< f | f eq X+1 or f eq X−1 where X is Parent(f).1 >;

  ClassParametersOPlusMinus := function( d, F )
    q := #F;
    if IsEven(q) then error "q must be odd"; end if;
    P<X> := PolynomialRing(F);
    pols := &cat[ All1IrreduciblePolynomialsSO(F, i) : i in [1. . d] ];
    parts := allPartitionsSHM(d);
    sparts := SignedPartitionsO( d );
    oldparams := [ [] : n in [1. . d] ];
    for f in pols do
        //print f, f;
      params := oldparams;
      fparts := IsSignedPol(f) select sparts else parts;
      for n in [0. . d−1] do
          //print n, n;
        dimleft := d−n;
        if Degree(f) le dimleft then
          multleft := dimleft div Degree(f);
          for i in [1. . multleft] do
            for param in ((n ne 0) select oldparams[n] else [[]]) do
              for part in fparts[i] do    //params; param cat [f,part];
                Append( ~params[n+Degree(f)∗i], param cat [<f, part>] );
              end for;
            end for;
          end for;
        end if;
      end for;
      oldparams := params;
    end for;
    return params[d];
  end function;

  testgo4 := procedure(d, q)
    printf "4. Class parameters for GO[0,1,-1](%o,%o)\n", d, q;
    time det := classInvariantsO(d, q);
    time shm, _ := ClassParametersOPlusMinus(d, GF(q));
    assert #det eq #shm;
    assert { Set(x) : x in det } eq { Set(y) : y in shm };
    print "Passed\n";
  end procedure;

  $testgo_4(4,5)$;
```

5. The Witt index calculation.

```
  WAdd := function( $q$, $w_1$, $w_2$ )
    if $q$ mod 4 eq 1 then
      return < $(w_1[1]+w_2[1])$ mod 2, $(w_1[2]+w_2[2])$ mod 2 >;
    else
      return < $(w_1[1]+w_2[1])$ mod 2, $(w_1[2]+w_2[2] + w_1[1]*w_2[1])$ mod 2 >;
    end if;
  end function;

  termToWitt := function( $f$, part, mult )
    $w$ := < Abs(part)*mult*Degree($f$) mod 2, 0 >;
    $P$<$X$> := Parent($f$);
    if IsDivisibleBy($X^2-1, f$) then
      $w[2]$ := (part gt 0) select 0 else 1;
    elif IsIrreducible($f$) then
      $w[2]$ := part*mult mod 2;
    end if;
    return $w$;
  end function;
```

The following function is a modified version of the function in `orthogonal.m`. In this case `param` is a sequence of pairs.

```
  paramToWitt := function( param )
    $f$ := param[1][1];
    $q$ := #BaseRing(Parent($f$));
    $w$ := <0,0>;
    for term in param do
      $f$ := term[1]; partition := term[2];
      if IsIrreducible($f$) then
        for $p$ in partition do    //termToWitt(f,p[1],p[2]);
          $w$ := WAdd($q, w$, termToWitt($f, p[1], p[2]$));
        end for;
      end if;
    end for;
    return $w$;
  end function;

  paramToSign := func< param | $(-1)^{\text{paramToWitt}(\text{param})}[2]$ >;

  ClassParametersGO := function( $d$, $F$ )
    error if IsEven($d$), "d should be odd";
    return [ param : param in ClassParametersOPlusMinus( $d$, $F$ ) |
      paramToSign(param) eq +1 ];
```

```
end function;

ClassParametersGOPlus := function( d, F )
  error if IsOdd(d), "d should be even";
  return [ param : param in ClassParametersOPlusMinus( d, F ) |
    paramToSign(param) eq +1 ];
end function;

ClassParametersGOMinus := function( d, F )
  error if IsOdd(d), "d should be even";
  return [ param : param in ClassParametersOPlusMinus( d, F ) |
    paramToSign(param) eq −1 ];
end function;

$testgo_5$ := procedure(d, q)
  printf "5. Class parameters for GO(%o,%o)\n", d, q;
  time det := ClassInvariantsGO(d, q);
  time shm, _ := ClassParametersGO(d, GF(q));
  assert #det eq #shm;
  assert { Set(x) : x in det } eq { Set(y) : y in shm };
  print "Passed\n";
end procedure;

testgo5p := procedure(d, q)
  printf "Class parameters for GOPlus(%o,%o)\n", d, q;
  time det := ClassInvariantsGOPlus(d, q);
  time shm, _ := ClassParametersGOPlus(d, GF(q));
  assert #det eq #shm;
  assert { Set(x) : x in det } eq { Set(y) : y in shm };
  print "Passed\n";
end procedure;

testgo5m := procedure(d, q)
  printf "Class parameters for GOMinus(%o,%o)\n", d, q;
  time det := ClassInvariantsGOMinus(d, q);
  time shm, _ := ClassParametersGOMinus(d, GF(q));
  assert #det eq #shm;
  assert { Set(x) : x in det } eq { Set(y) : y in shm };
  print "Passed\n";
end procedure;

$testgo_5$(5, 5);
testgo5p(6, 5);
testgo5m(6, 5);
```

6. Type 1 companion matrices

Given an irreducible polynomial $f(t)$ such that $f^*(t) \neq f(t)$ and a partition $\pi = [\langle 1, m_1 \rangle, \ldots, \langle k, m_k \rangle]$ we want the matrix corresponding to $\langle f(t) f^*(t), \pi \rangle$.

First of all we set up Scott's code. Before introducing CMatO we need a few supporting functions.

Find x in $K = \text{ext}\langle F \mid 2\rangle$ such that $a(x^2 + x^{2q}) = bN(x)$.

```
NTQuadEq := function( K, F, a, b )
  if a eq 0 then
    return (b eq 0) select 1 else K ! 0;
  elif b+2*a eq 0 then
    return Sqrt(K ! NonSquare(F));
  elif IsSquare((b−2*a)/(b+2*a)) then
    q := #F;
    if exists(c){ c : c in F | c ne 0 and a*(c^2+c^(2*q)) eq b*c^(q+1)} then
      return c;
    else
      error "This is a particularly deliquescent bug in Magma.
      Please email the run to murray@maths.usyd.edu.au";
    end if;
      // this never happens for us, I don't know why
  else
    P<X> := PolynomialRing(K);
    rts := Roots( X^2−X+a/(b+2*a) );
    c := rts[1][1];
    return c;
  end if;
end function;

nufunc := function( k, K )
  q := #k;
  ξ := PrimitiveElement( K );
  return Sqrt( k ! ( 4*ξ^(q+1)/(ξ−ξ^q)^2 ) );
end function;
```

The parameter t is a field element and f is either an odd power of a polynomial of type 2 or a power of a polynomial of type 1. If $t = 1$ and f is of type 2, CMatO should return an element in $\text{GO}^-(d, q)$.

```
CMatO := function( t, f )
  k := BaseRing( f ); K := ext<k|2>; q := #k;
  d := Degree( f ); square_t := IsSquare(t);
  P<X> := Parent( f );

  fact := Factorisation(f);
  if #fact eq 2 then
    g := fact[1][1]^(fact[1][2]);
    C := CompanionMatrix(g);
    F := SymmetricBilinearForm(+1, d div 2, k);
    return DiagonalJoin( C, t*F*Transpose(C)^−1*F );
  end if;
```

```
$\delta$ := NONSQUARE($k$); $\delta_0$ := SQRT($K$ ! $\delta$);
if $d$ eq 1 then
   return MATRIX( 1,1, [$k$| −COEFFICIENT($f$,0)] );
elif $f$ eq $X^2-t$ then
   $\nu$ := nufunc($k$,$K$);
   $x$ := SQRT($t*(1+\nu^2)$); $y$ := SQRT($t/\delta$);
   return MATRIX( 2,2, [ $k$| $x$, $\nu*y$, $-\nu*\delta*y$, $-x$ ] );
elif $d$ eq 2 then
   $a$ := COEFFICIENT($f$,1); disc := $a^2-4*t$;
   $x$ := SQRT($\delta$*disc);
   return MATRIX( 2,2, [ $k$| $-a/2$, $x/(2*\delta)$, $x/2$, $-a/2$ ] );
end if;
$r$ := $d$ div 2 −1;
as := COEFFICIENTS( $f$ )[1. . $r$+2]; a:= func< i | as[i+1] >;

   // compute c
st := &+[ k | $a(r-2*i)/t^i$ : i in [0. . (r div 2)] ];
sn := ( $a(r+1)$ + 2∗ &+[ k | $a(r+1-2*i)/t^i$ : i in [1. . ((r+1) div 2)] ] );

if square_t then
   srt := SQRT($t$);
   c := NTQUADEQ($K$,$k$,st,sn∗srt);

   assert $st*(c^2+c^{(2*q)})$ eq $sn*srt*c^{(q+1)}$;
   assert $st*srt*(c^2+c^{(2*q)})$ eq $sn*t*c^{(q+1)}$;
   bp := k ! 0;
else
   if exists(pair){ <bp,c> : bp in k, c in K | c ne 0 and
      ISSQUARE($\delta*bp^2+t$) and
      ( $t*sn*c^{(q+1)}$ eq $st*(srt*(c^2+c^{(2*q)}) + bp*\delta_0*(c^{(2*q)}-c^2))$
         where srt is SQRT($\delta*bp^2+t$) ) }
   then
      bp := pair[1]; c := pair[2]; srt := SQRT($\delta*bp^2+t$);
   else
      error “This is a particularly deliquescent bug in Magma.
      (2) Please email the run to murray@maths.usyd.edu.au”;
   end if;
end if;
assert $srt^2$ eq $\delta*bp^2+t$;
assert $t*sn*c^{(q+1)}$ eq $st*(srt*(c^2+c^{(2*q)}) + bp*\delta_0*(c^{(2*q)}-c^2))$;

   // compute b_i for i=0..r-1
bs := [ (st ne 0) select $t^{(r+1)}*c^{(q+1)}/st$ else k ! 2 ];
b:= func< i | bs[i+1] >;
for i in [1. . r−1] do
   bs[i+1] := ( $a(i)*b(0)$ + ((i ne 1) select $t^r*b(i-2)$ else 0) )/$t^{(r+1)}$;
   b:= func< i | bs[i+1] >;
```

```
end for;

cs := c^q;
assert Coefficient(f, 0) eq t^(r+1);
for i in [1..r-1] do
   assert Coefficient(f, i) eq
      (t^(r+1)*b(i) - ((i ne 1) select t^r*b(i-2) else 0))/b(0);
end for;
assert r lt 2 or
   Coefficient(f, r) eq (t^(r+1)*c*cs - ((r ne 1) select t^r*b(r-2) else 0))/b(0);

   // construct the matrix
C := ZeroMatrix( k, d, d );
for i in [1..r-1] do
   C[i, i+1] := 1;
end for;

if r gt 0 then C[r, d] := 1/b(0); end if;

C[r+1, r+1] := srt; C[r+1, r+2] := -bp;
C[r+1, d] := -(-1)^r * (bp*(c-cs)*δ_0 - srt*(c+cs)) /(2*b(0));

C[r+2, r+1] := bp*δ; C[r+2, r+2] := -srt;
C[r+2, d] := -(-1)^r * (srt*δ_0*(c-cs) - bp*δ*(c+cs)) /(2*b(0));

for i in [1..r] do C[d-r+1, i] := t*b(i-1); end for;
if r gt 0 then
   C[d-r+1, r+1] := -(-1)^r*t*(c+cs); C[d-r+1, r+2] := (-1)^r*t*(c-cs)/(δ_0);
   C[d-r+1, d] := - t*c*cs/b(0);
end if;

for i in [2..r] do
   C[d-r+i, d-r+i-1] := t;
   C[d-r+i, d] := -t*b(r+1-i)/b(0);
end for;

F := SymmetricBilinearFormMinus(d, k);
F_2 := DiagonalMatrix( [ k | 1/2, -δ/2 ] );
InsertBlock(~F, F_2, r+1, r+1);
assert C*F*Transpose(C) eq t*F;

   // Convert to standard form
_, T_2 := TransformBilinearForm( F_2 );
I := IdentityMatrix( k, r );
T := DiagonalJoin( I, DiagonalJoin( T_2, I ) );
C := T*C*T^-1;

   // check
F := SymmetricBilinearFormMinus(d, k);
```

```
    assert C*F*Transpose(C) eq t*F;
    assert CharacteristicPolynomial(C) eq f;

    return C;
end function;

JMatL := function( C, m )
    F := BaseRing( C );
    if m eq 0 then return Matrix(0,0,[F|]); end if;
    d := Nrows( C );
    J := DiagonalJoin( [Parent(C)|C: i in [1..m]] );
    I := Parent(C) ! 1;
    for i in [1..m-1] do
        InsertBlock( ~J, I, (i-1)*d+1, i*d+1 );
    end for;
    return J;
end function;

JMatOOddOld := function( t, f, part )
    F := BaseRing( f ); P<X> := Parent(f);
    sgn := Sign( part );
    m := Abs(part) div 2;
    d := Degree(f); Id := IdentityMatrix(F, d);
    C := CompanionMatrix(f);
    J := JMatL( Matrix(1,1,[F ! 1]), m );
    X := DiagonalJoin( J, DiagonalJoin( Matrix(1,1,[F ! 1]), J^-1 ));
    if Abs(part) gt 1 then X[m,m+1] := 1; end if;
    X := KroneckerProduct(X,C);
    if sgn eq +1 then
        a := Id;
    elif d eq 1 then
        a := Nonsquare(F)*Id;
    else
        E := ext< F | f >;
        a := WriteOverSmallerField( Matrix(1,1,[Nonsquare(E)]), F);
    end if;

    for i in [1..m] do
        InsertBlock( ~X, (-1)^i*a*C/2, (m-1)*d+1,(m+i)*d+1 );
    end for;
    if sgn eq -1 and d eq 2 then
        C := CMatO(t,f);
        InsertBlock( ~X, C, m*d+1, m*d+1 );
    end if;
    for i in [1..m] do
        InsertBlock( ~X, (-1)^i*a*C, m*d+1,(m+i)*d+1 );
    end for;
/*
```

```
  assert IsScalar( X*J*Transpose(X) * J^-1 ) where
    J is SymmetricBilinearForm( sgn, Abs(part), F );
*/

  return X;
end function;

JMatOOdd := function(t, f, part)
  F := BaseRing( f ); P<X> := Parent(f);
  sgn := Sign( part );
  if sgn eq −1 then
    return OrthogonalTensorProduct(CMatO(t, f), JMatOOddOld(F ! 1, X−1, part):
      ASign:=−1, BSign:=−1, Sign:=−1);
  else
    return OrthogonalTensorProduct(CompanionMatrix(f),
      JMatOOddOld(F ! 1, X−1, part): ASign:=+1, BSign:=+1, Sign:=+1);
  end if;
end function;

JMatOUnsigned := function(f, part)
  k := BaseRing( f ); q := #k;
  d := Degree( f ); P<X> := Parent( f );
  fact := Factorisation(f);
  if #fact eq 2 then
    g := $fact[1][1]^{fact[1]}[2]$;
    C := JMatL(CompanionMatrix(g), part);
    F := SymmetricBilinearForm(+1, (d div 2)∗part, k);
    return DiagonalJoin( C, $F*\text{Transpose}(C)^{-1}*F$ );
  elif IsOdd(part) then
    return OrthogonalTensorProduct(CMatO(k ! 1, f), JMatOOddOld(k ! 1, X−1, part) :
      ASign:=−1, BSign:=+1, Sign:=$(-1)^{part}$);
  else
    sd := d∗(part div 2);
    C := $$(f, part div 2);
    $F_1$ := SymmetricBilinearForm($(-1)^{(part\ div\ 2)}$, sd, k);
    assert $C*F_1*$Transpose(C) eq $F_1$;
    $F_0$ := SymmetricBilinearForm(+1, sd, k);
    Bt := ZeroMatrix(k, sd, sd);
    Bt[sd, sd−1] := −1; Bt[sd−1, sd] := 1;
    B := $\text{Bt}*(F_1*\text{Transpose}(C))^{-1}$;
    A := BlockMatrix(2, 2, [C, B, 0, C]);
    assert A∗F∗Transpose(A) eq F where F is BlockMatrix(2, 2, [0, $F_1$, $F_1$, 0]);;
    T := BlockMatrix(2, 2, [$F_0*F_1^{-1}$, 0, 0, 1]);
    return $T*A*T^{-1}$;
  end if;
end function;

wittToSign := func< w | $(-1)^{w[2]}$ >;
```

```
UnsignedHMatO := function( f, partition )
   F := BaseRing(Parent(f));
   q := #F;
   X := ZeroMatrix( F, 0, 0 ); w := <0,0>;
   for part in partition do
      J := JMatOUnsigned( f, part[1] );
      wJ := paramToWitt( [ < f, [<part[1],1>] > ] );
      for i in [1..part[2]] do
         oldw := w; w := WAdd(q, w, wJ);    // oldw,wJ,w;
         X := OrthogonalDirectSum( X, J :
            ASign:=wittToSign(oldw), BSign:=wittToSign(wJ), Sign:=wittToSign(w) );
      end for;
   end for;
   return X;
end function;

testgo6 := procedure(q)
   print "6. Type 1 companion matrices";
   F := GF(q);
   _<x> := PolynomialRing(F);
   f := x^3 + 2*x +1;
   h := f*DualPolynomial(f);
   A := type1Companion(f^2);
   B := JMatOUnsigned(h,2);
   J := StandardSymmetricForm(12,F);
   assert A*J*Transpose(A) eq J;
   assert B*J*Transpose(B) eq J;
   assert PrimaryInvariantFactors(A) eq PrimaryInvariantFactors(B);
   print "Passed\n";
end procedure;

testgo6s := procedure(q)
   print "6s. Type 1 companion matrix sequence";
   F := GF(q);
   _<x> := PolynomialRing(F);
   f := x^3 + 2*x +1;
   h := f*DualPolynomial(f);
   π := [<1,2>,<2,1>];
   A := type1Matrix(h,π);
   B := UnsignedHMatO(h,π);
   n := Nrows(A);
   J := StandardSymmetricForm(n,F);
   assert A*J*Transpose(A) eq J;
   assert B*J*Transpose(B) eq J;
   assert PrimaryInvariantFactors(A) eq PrimaryInvariantFactors(B);
   print "Passed\n";
end procedure;
```

```
testgo_6(5);
testgo6s(9);
```

7. Type 1 and 2 companion matrices

```
testgo_7 := procedure(d, q)
  d := 6;
  printf "7. Type 1 and 2 companion matrices for GOMinus(%o,%o)\n",
   d, q;
  invs := ClassInvariantsGOMinus(d, q);
  Z := [ z : z in invs | forall{ w : w in z | Degree(w[1]) gt 1} ];
  print #Z;
  time mats := [RepresentativeMatrixO($\mu$) : $\mu$ in Z ];

  printf "Type 1 and 2 companion matrices for GOPlus(%o,%o)\n",
    d, q;
  invs := ClassInvariantsGOPlus(d, q);
  Z := [ z : z in invs | forall{ w : w in z | Degree(w[1]) gt 1} ];
  print #Z;
  time mats := [RepresentativeMatrixO($\mu$) : $\mu$ in Z ];

  print "Passed\n";
end procedure;

testgo_7(6, 5);
```

8. Type 3 companion matrices

```
testgo_8 := procedure(d, q)
  d := 6;
  printf "8. Type 3 companion matrices for GOMinus(%o,%o)\n",
   d, q;
  invs := ClassInvariantsGOMinus(d, q);
  Z := [ z : z in invs | forall{ w : w in z | Degree(w[1]) eq 1} ];
  print #Z;
  time mats := [RepresentativeMatrixO($\mu$) : $\mu$ in Z ];

  printf "8. Type 3 companion matrices for GOPlus(%o,%o)\n",
    d, q;
  invs := ClassInvariantsGOPlus(d, q);
  Z := [ z : z in invs | forall{ w : w in z | Degree(w[1]) eq 1} ];
  print #Z;
  time mats := [RepresentativeMatrixO($\mu$) : $\mu$ in Z ];

  print "Passed\n";
end procedure;

testgo_8(6, 5);
```

9. Compute conjugacy invariants from given matrices.

```
$testgo_9$ := procedure($d, q$ : Minus := false)
  if IsOdd($d$) then
    printf "9. Conjugacy invariants for GO(%o,%o)\n", $d$, $q$;
    $G$ := GO($d, q$);
    $Y$ := ClassInvariantsGO($d, q$);
  else
    $tag$ := Minus select "Minus" else "Plus";
    printf "9. Conjugacy invariants for GO%o(%o,%o)\n", $tag$, $d$, $q$;
    $G$ := Minus select GOMinus($d, q$) else GOPlus($d, q$);
    $Y$ := Minus select ClassInvariantsGOMinus($d, q$)
         else ClassInvariantsGOPlus($d, q$);
  end if;
  $X$ := [ConjugacyInvariantO($c$[3] : Minus := Minus) : $c$ in Classes($G$)];
  assert Set($X$) eq Set($Y$);
  print "Passed\n";
end procedure;

$testgo_9$(3, 3);
$testgo_9$(5, 5);
$testgo_9$(4, 3);
$testgo_9$(4, 3 : Minus);
$testgo_9$(6, 5);
$testgo_9$(6, 5 : Minus);
```

10. Centraliser orders from conjugacy class invariants

```
$testgo_{10}$ := procedure($d, q$ : Minus := false)
  if IsOdd($d$) then
    printf "10. Centraliser orders for GO(%o,%o)\n", $d$, $q$;
    $G$ := GO($d, q$);
  else
    $tag$ := Minus select "Minus" else "Plus";
    printf "10. Centraliser orders for GO%o(%o,%o)\n", $tag$, $d$, $q$;
    $G$ := Minus select GOMinus($d, q$) else GOPlus($d, q$);
  end if;
  $cc$ := Classes($G$);
  $ord$ := #$G$;
  $X$ := [ConjugacyInvariantO($c$[3] : Minus := Minus) : $c$ in $cc$];
  assert [CentraliserOrderO($x$) : $x$ in $X$] eq [$ord$ div $c$[2] : $c$ in $cc$];
  print "Passed\n";
end procedure;

$testgo_{10}$(3, 5);
$testgo_{10}$(4, 5);
$testgo_{10}$(4, 5 : Minus);
```

11. Class representatives

```
$testgo_{11}$ := procedure($d, q$ : Minus := false)
```

```
    if IsOdd($d$) then
      printf "11. Class representatives for GO(%o,%o)\n", $d$, $q$;
      $G$ := GO($d, q$);
      reps := [$G$| RepresentativeMatrixO($\mu$) : $\mu$ in ClassInvariantsGO($d, q$)];
    else
      tag := Minus select "Minus" else "Plus";
      printf "11. Class representatives for GO%o(%o,%o)\n", tag, $d$, $q$;
      $G$ := Minus select GOMinus($d, q$) else GOPlus($d, q$);
      reps := Minus select ClassRepresentativesGOMinus($d, q$)
                  else ClassRepresentativesGOPlus($d, q$);
    end if;
    cc := Classes($G$);
    ndx := {};
    for $X$ in reps do
      assert exists($i$){ $i$ : $i$ in [1..#cc] | IsConjugate($G, X, cc[i][3]$) };
      Include(~ndx, $i$);
    end for;
    assert #reps eq #ndx;
    print "Passed\n";
  end procedure;

  $testgo_{11}(4, 3)$;
  $testgo_{11}(4, 3$ : Minus);
  $testgo_{11}(3, 3)$;
```

12. Conjugacy invariants (randomised)

```
  $testgo_{12}$ := procedure($n, r$ : Minus := false)
    if IsOdd($n$) then
      printf "12. Randomised conjugacy invariants for GO(%o,%o)\n",
        $n, r$;
      $G$ := GO($n, r$);
      $X$ := ClassInvariantsGO($n, r$);
    else
      tag := Minus select "Minus" else "Plus";
      printf "12. Randomised conjugacy invariants for GO%o(%o,%o)\n",
        tag, $n$, $r$;
      $G$ := Minus select GOMinus($n, r$) else GOPlus($n, r$);
      $X$ := Minus select ClassInvariantsGOMinus($n, r$) else ClassInvariantsGOPlus($n, r$);
    end if;
    for $\mu$ in $X$ do
      $g$ := RepresentativeMatrixO($\mu$);
      $h$ := Random($G$);
      $c$ := ConjugacyInvariantO($g^h$ : Minus := Minus);
      assert $\mu$ eq $c$;
    end for;
    print "Passed\n";
  end procedure;
```

```
testgo12(4, 9) ;
testgo12(6, 5 : Minus) ;
```

References

[1] W. Bosma, J. Cannon, C. Fieker, and A. Steel. *Handbook of Magma functions*. University of Sydney, v2.21 edition, May 2015.

[2] W. Bosma, J. Cannon, and C. Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997. Computational algebra and number theory (London, 1993).

[3] J. R. Britnell. *Cycle index methods for matrix groups over finite fields*. DPhil Thesis, University of Oxford, 2003.

[4] J. Fulman. *Probability in the Classical Groups over Finite Fields: Symmetric Functions, Stochastic Algorithms, and Cycle Indices*. PhD Thesis, Harvard University, 1997.

[5] J. Fulman. A probabilistic approach to conjugacy classes in the finite symplectic and orthogonal groups. *J. Algebra*, 234(1):207–224, 2000.

[6] S. Haller and S. H. Murray. Computing conjugacy in finite classical groups 1: similarity in unitary groups. preprint, January 2009.

[7] W. H. Hesselink. Nilpotency in classical groups over a field of characteristic 2. *Math. Z.*, 166(2):165–181, 1979.

[8] J. E. Humphreys. *Conjugacy classes in semisimple algebraic groups*, volume 43 of *Mathematical Surveys and Monographs*. American Mathematical Society, Providence, RI, 1995.

[9] M. Knebusch and M. Kolster. *Wittrings*, volume 2 of *Aspects of Mathematics*. Friedr. Vieweg & Sohn, Braunschweig, 1982.

[10] M. W. Liebeck and G. M. Seitz. *Unipotent and nilpotent classes in simple algebraic groups and Lie algebras*, volume 180 of *Mathematical Surveys and Monographs*. American Mathematical Society, Providence, RI, 2012.

[11] I. G. Macdonald. *Symmetric functions and Hall polynomials*. Oxford Mathematical Monographs. The Clarendon Press Oxford University Press, New York, second edition, 1995. With contributions by A. Zelevinsky, Oxford Science Publications.

[12] J. Milnor. On isometries of inner product spaces. *Invent. Math.*, 8:83–97, 1969.

[13] S. H. Murray. Computing conjugacy in finite classical groups 2: similarity in symplectic and orthogonal groups. preprint, July 2007.

[14] K. Shinoda. The characters of Weil representations associated to finite fields. *J. Algebra*, 66(1):251–280, 1980.

[15] T. A. Springer and R. Steinberg. Conjugacy classes. In *Seminar on Algebraic Groups and Related Finite Groups (The Institute for Advanced Study, Princeton, N.J., 1968/69)*, Lecture Notes in Mathematics, Vol. 131, pages 167–266. Springer, Berlin, 1970.

[16] D. E. Taylor. *The Geometry of the Classical Groups*, volume 9 of *Sigma Series in Pure Mathematics*. Heldermann Verlag, Berlin, 1992.

[17] G. E. Wall. On the conjugacy classes in the unitary, symplectic and orthogonal groups. *J. Aust. Math. Soc.*, 3:1–62, 1963.

[18] J. Williamson. On the normal forms of linear canonical transformations in dynamics. *Amer. J. Math.*, 59(3):599–617, 1937.

[19] E. Witt. Theorie der quadratischen Formen in beliebigen Körpern. *J. Reine Angew. Math.*, 176:31–44, 1937.

[20] T. Xue. Nilpotent orbits in classical Lie algebras over finite fields of characteristic 2 and the Springer correspondence. *Represent. Theory*, 13:371–390, 2009.