

# Constructing the group preserving a system of forms

Peter A. Brooksbank   E.A. O'Brien

## Abstract

We present a practical algorithm to construct the subgroup of the general linear group that preserves a system of bilinear or sesquilinear forms on a vector space defined over a finite field. Components include efficient algorithms to construct the Jacobson radical and the group of units of a matrix algebra.

## 1 Introduction

In this paper we consider the following algorithmic problem: *Given a system of symmetric, alternating or hermitian forms on a vector space defined over a finite field, find the subgroup of the general linear group that preserves every form in the system.*

In [2], an explicit description of this group is given when the system contains precisely two forms of the same classical type, at least one of which is nondegenerate. This description, heavily influenced by the recent work of Goldstein & Guralnick [8], underpins a highly effective algorithm to write down generators for the group. The structure of the group preserving a general system of forms is much more varied, and consequently the algorithm for the general case is more limited.

The motivation for an efficient solution to the general problem of constructing the group preserving a system of forms is two-fold.

First, as noted in [2], constructing the intersection of two or more matrix groups is a difficult algorithmic problem. Existing algorithms employ variations of “back-track”, their complexity is exponential, and they have very limited range; see [14, §3.3] for a general discussion.

---

The first author thanks the Department of Mathematics at the University of Auckland for its hospitality while this work was done. This work was supported in part by the Marsden Fund of New Zealand via grant UOA 412. We thank Lajos Rónyai for drawing our attention to [13] and for comments on a draft of the paper, and Derek Holt for assistance with our implementation. 2000 *Mathematics Subject Classification*. Primary 20C20, 20C40.

One outcome of this work is a significantly better algorithm to construct the intersection of a collection of matrix groups that preserve forms on the underlying space.

The group preserving a system of forms also arises naturally in the study of finite  $p$ -groups. To construct the automorphism group of a  $d$ -generator  $p$ -group of exponent- $p$ -class 2 and exponent  $p$ , we consider the action of the general linear group  $\mathrm{GL}(V)$  on the exterior square  $\Lambda^2(V)$  of  $V = \mathrm{GF}(p)^d$  and compute the stabiliser of a subspace in this action. A basis of a  $k$ -dimensional subspace  $W$  of  $\Lambda^2(V)$  defines a system  $\Sigma(W)$  consisting of  $k$  alternating forms on  $V$ . The subgroup of  $\mathrm{GL}(V)$  preserving  $\Sigma(W)$  is the centraliser of  $W$ , a normal subgroup of the desired stabiliser. For further discussion of this application, see [6].

As a key component of this work, we describe effective algorithms to construct both the Jacobson radical and the group of units of a matrix algebra defined over a finite field. We use the unit group algorithm to write down an overgroup of  $I$ , the group preserving the given system of forms, and then use a variety of techniques to construct  $I$  within this overgroup.

Our approach to the unit group problem was motivated by work of Schwingel [15]. In her PhD thesis she describes an algorithm to construct the subgroup of  $\mathrm{GL}(V)$  that stabilises a lattice of subspaces of  $V$ ; it is obtained as the group of units of the algebra stabilising the lattice. While writing this paper, we learned that Rónyai [13], in solving a complexity question about the orders of centralisers in the general linear group, had earlier suggested an identical approach to the unit group problem.

We present optimised versions of the algorithms, aimed at practical implementation. We demonstrate that our implementation of the Jacobson radical algorithm significantly outperforms the algorithm of Cohen *et al.* [3], which applies to arbitrary fields.

The unit group algorithm is presented in Section 2, where we also identify those steps comprising the Jacobson radical algorithm. In Section 3 we describe how to construct bases for two important families of matrix algebras. The algorithm to construct the (full isometry) group preserving a system of forms is described in Section 4, together with a summary of the modifications needed to obtain the intersection of a collection of groups preserving such a system. Finally, in Section 5, we briefly describe an implementation of our methods in MAGMA [1], indicating their practical limitations, and reporting on their performance.

## 2 The group of units of a matrix algebra

Let  $\mathbb{M}(d, F)$  denote the algebra of all  $d \times d$  matrices with entries in the finite field  $F$ , and let  $A$  be a subalgebra of  $\mathbb{M}(d, F)$ . We present an efficient algorithm to write down generators for  $U(A)$ ,

the group of units of  $A$ . We assume that  $A$  is described by  $X \subset \mathbb{M}(d, F)$ . Hence  $A = \text{Env}(X)$ , the enveloping algebra of  $X$ , defined formally as the  $F$ -linear span of the semigroup generated by  $X$ .

In Section 2.1 we recall Wedderburn's structure theorem for semisimple algebras [4, 26.4], and its relationship to a composition series of the natural  $A$ -module  $V = F^d$ . This provides the theoretical foundation for the algorithm, which is described in detail in Section 2.2.

## 2.1 Composition series and the Wedderburn decomposition

The Jacobson radical of  $A$ , denoted  $J(A)$ , is the intersection of the maximal left ideals of  $A$ , and the quotient algebra  $\bar{A} = A/J(A)$  is semisimple. If  $F$  is a finite field, then Wedderburn's theorem states that  $\bar{A} = B_1 \oplus \dots \oplus B_r$ , where  $B_i$  is a minimal left ideal of  $\bar{A}$ , isomorphic as  $F$ -algebra to  $\mathbb{M}(d_i, F_i)$  for some extension field  $F_i$  of  $F$ . Note that  $U(\bar{A})$ , the unit group of  $\bar{A}$ , is isomorphic to  $\text{GL}(d_1, F_1) \times \dots \times \text{GL}(d_r, F_r)$ .

The natural  $A$ -module  $V = F^d$  has a composition series  $0 = V_0 < V_1 < \dots < V_s = V$  and, relative to a suitable basis of  $V$ , an element  $a$  of  $A$  has the form

$$\begin{bmatrix} a_1 & \star & \dots & \dots & \star \\ 0 & a_2 & \star & \dots & \star \\ \vdots & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & a_{s-1} & \star \\ 0 & 0 & \dots & 0 & a_s \end{bmatrix}. \quad (1)$$

Define an equivalence relation on the indices  $\{1, \dots, s\}$ , where  $i \sim j$  if and only if the quotients  $V_i/V_{i-1}$  and  $V_j/V_{j-1}$  are isomorphic  $A$ -modules. Denote the equivalence classes by  $\Gamma_1, \dots, \Gamma_t$ , and let  $e_i$  be the common dimension of the quotient spaces indexed by elements of  $\Gamma_i$ . For  $1 \leq i \leq t$ , distinguish an index  $\gamma_i \in \Gamma_i$ . For each  $\gamma \in \Gamma_i$ , there exists  $c_\gamma \in \text{GL}(e_i, F)$  such that

$$a_\gamma = c_\gamma a_{\gamma_i} c_\gamma^{-1}. \quad (2)$$

For  $1 \leq i \leq s$ , let  $A_i$  be the algebra induced by  $A$  on the quotient  $V_i/V_{i-1}$ . Now  $A_i$  is a simple matrix algebra and is therefore isomorphic to a full matrix algebra over some extension field of  $F$ . The map  $a \mapsto (a_1, a_2, \dots, a_{s-1}, a_s)$  is a homomorphism  $\varphi: A \rightarrow A_1 \oplus \dots \oplus A_s$ , whose image is isomorphic to  $\bigoplus_{i=1}^t A_{\gamma_i}$ . Since  $\varphi(A)$  is semisimple, and  $\ker(\varphi)$  is nilpotent,  $\ker(\varphi) = J(A)$  and  $\varphi(A) \cong \bar{A}$ . In particular,  $r = t$  and, up to permutation of indices,  $A_{\gamma_i}$  is isomorphic to  $B_i$ .

## 2.2 The Jacobson radical and unit group algorithms

In outline, the algorithm to construct generators for  $U(A)$  is the following:

- Step 1.** Construct an algebra epimorphism  $\varphi: A \rightarrow B$  whose kernel is  $J(A)$ , and obtain the Wedderburn decomposition of the semisimple algebra  $B$ .
- Step 2.** Use the Wedderburn decomposition to write down a generating set for  $U(B)$ .
- Step 3.** Compute a preimage,  $S$ , of this generating set under  $\varphi$ .
- Step 4.** Compute a generating set  $T$  for the kernel of the induced map  $\varphi_U: U(A) \rightarrow U(B)$ .
- Step 5.** Return  $S \cup T$ .

There are two important algorithmic issues to address. The first is how to construct  $\varphi$  and obtain the Wedderburn decomposition of  $B$  into simple algebras. To do this, we exploit the composition series of the natural  $A$ -module  $V$ , as discussed in Section 2.1. The second is how to define  $\varphi$  *effectively*; in particular, an efficient procedure is required to compute the preimage of an element of  $B$  under  $\varphi$ .

We begin by defining a matrix algebra  $A$  as the solution to a system of linear equations. Define an isomorphism of vector spaces  $\rho: \mathbb{M}(d, F) \rightarrow F^{d^2}$  sending  $[[x_{ij}]] \mapsto (x_{11}, x_{12}, \dots, x_{dd})$ . Suppose that  $A \leq \mathbb{M}(d, F)$  is a  $k$ -dimensional algebra having basis  $b_1, \dots, b_k$ . Define a  $k \times d^2$  matrix  $\mathcal{B}$  whose rows are the vectors  $\rho(b_1), \dots, \rho(b_k)$ . Then  $\mathcal{B}$  is a (left)  $F$ -linear map from the column space  $F^{d^2}$  to the column space  $F^k$ ; let  $\mathcal{A}$  be a  $d^2 \times (d^2 - k)$  matrix whose columns are a basis for the nullspace of this map. For  $x \in \mathbb{M}(d, F)$ , we have the following test for membership in  $A$ :

$$x \in A \iff \rho(x)\mathcal{A} = 0. \quad (3)$$

Next suppose that we have found a composition series for the natural module of a given algebra  $A = \text{Env}(X) \leq \mathbb{M}(d, F)$ . Suppose further that we have a change-of-basis matrix that conjugates the elements of  $A$  to matrices of the form in Equation (1). This defines a homomorphism  $\varphi: A \rightarrow A_1 \oplus A_2 \oplus \dots \oplus A_s$ , where the algebras  $A_i$  are as defined in Section 2.1.

The following result enables us to compute effectively with the homomorphism  $\varphi$ .

**Lemma 2.1** *There are deterministic algorithms using  $O(d^6)$  field operations to solve each of the following.*

- (i) Given  $b = (a_1, a_2, \dots, a_s) \in \varphi(A)$ , find  $a \in A$  such that  $\varphi(a) = b$ .

(ii) Find a basis for  $\ker(\varphi)$ .

(iii) Find a generating set for the kernel of the induced map  $\varphi_U: U(A) \rightarrow U(\varphi(A))$ .

**Proof.** (i) Let  $b = (a_1, \dots, a_s)$  be a given element of  $\varphi(A)$  and suppose, for  $1 \leq l \leq s$ , that  $a_l$  is an  $n_l \times n_l$  matrix with entries in  $F$ . Set  $n := \sum_{l=1}^s n_l^2$ . We augment  $\mathcal{A}$  to a  $(d^2 + 1) \times (d^2 - k + n)$  matrix  $\mathcal{A}^{(b)}$  and construct a suitable preimage of  $b$  from the nullspace of this augmented matrix.

Initialise  $\mathcal{A}^{(b)}$  to be the  $(d^2 + 1) \times (d^2 - k)$  matrix obtained by appending a row of zeros to  $\mathcal{A}$ . For  $z \in [1, \dots, d^2 + 1]$  and scalar  $\lambda \in F$ , define a column vector  $c(z, \lambda) \in F^{d^2+1}$  as follows: if  $\lambda = 0$ , then  $c(z, \lambda)$  has 1 in coordinate  $z$  and 0s in all other coordinates; and if  $\lambda \neq 0$ , then  $c(z, \lambda)$  has  $-\lambda$  in coordinate  $z$ , 1 in coordinate  $d^2 + 1$ , and 0s in all other coordinates. The following loop now appends the additional  $n$  columns to  $\mathcal{A}^{(b)}$ .

```

For  $l \in \{1, \dots, s\}$  do
     $m_l := n_1^2 + \dots + n_l^2$ 
    For  $i, j \in \{1, \dots, n_l\}$  do
         $z := m_l + (i - 1)n_l + j$ ;  $\lambda := (a_l)_{ij}$ 
        Append  $c(z, \lambda)$  to  $\mathcal{A}^{(b)}$ 

```

For a row vector  $y \in F^{d^2}$ , let  $y^*$  denote the row vector in  $F^{d^2+1}$  obtained from  $y$  by appending 1 in coordinate  $d^2 + 1$ . An easy calculation now shows that  $a \in \mathbb{M}(d, F)$  satisfies  $\rho(a)^* \mathcal{A}^{(b)} = 0$  if and only if  $a \in A$  and  $\varphi(a) = b$ .

To find a preimage of the given  $b = (a_1, \dots, a_s)$  in  $A$ , proceed as follows:

- (a) Compute the augmented matrix  $\mathcal{A}^{(b)}$  associated with  $b$ , as above.
- (b) Compute a basis for the nullspace of  $\mathcal{A}^{(b)}$ .
- (c) Find a vector  $(\lambda_1, \dots, \lambda_{d^2}, \lambda)$  in this basis having  $\lambda \neq 0$ .
- (d) Put  $y := (\lambda_1/\lambda, \dots, \lambda_{d^2}/\lambda) \in F^{d^2}$ , and return  $a := \rho^{-1}(y)$ .

(ii) To construct the kernel of  $\varphi$  we proceed in a similar fashion, this time augmenting  $\mathcal{A}$  to a  $d^2 \times (d^2 - k + n)$  matrix  $\mathcal{A}^{\ker}$ . We wish to compute the full preimage in  $A$  of the zero element of  $B$ , namely  $(0_{A_1}, 0_{A_2}, \dots, 0_{A_s})$ . Thus, for each  $1 \leq l \leq s$  and  $1 \leq i, j \leq n_l$ , append a column to  $\mathcal{A}$  having 1 in coordinate  $m_l + (i - 1)n_l + j$  and 0 in all other coordinates.

The kernel of  $\varphi$  is now obtained as follows:

- (a) Compute the augmented matrix  $\mathcal{A}^{\ker}$  as above.

(b) Compute a basis,  $\mathcal{B}$ , for the nullspace of  $\mathcal{A}^{\ker}$ .

(c) Return the set  $\{\rho^{-1}(y) : y \in \mathcal{B}\}$ .

(iii) Using the algorithm in (ii), obtain an  $F$ -basis for  $J := \ker(\varphi)$ , the Jacobson radical of  $A$ . Convert this  $F$ -basis into a basis over the prime field as follows. Let  $e$  denote the degree of  $F$  over its prime field, and let  $\zeta$  be a primitive element of  $F^*$ ; now compute  $\rho^{-1}(\zeta^j y)$  for  $y \in \mathcal{B}$  and  $j \in [0, \dots, e-1]$ . Next exchange this arbitrary basis for another basis,  $T^*$ , over the prime field containing elements that project onto bases for the layers  $J^i/J^{i+1}$  for  $1 \leq i \leq d$ . (This is a linear algebra computation in  $J$  regarded as a vector space over the prime field.) Let  $I_d$  denote the identity of  $\mathbb{M}(d, F)$ ; then it is easy to see that the set

$$T := \{I_d + u : u \in T^*\}, \quad (4)$$

generates the kernel of  $\varphi_U$ .

The stated complexity is the time required to perform basic linear algebra in  $F^{d^2}$ .  $\square$

We now give a detailed description of the algorithm to construct the group of units of a matrix algebra. The basic steps are numbered consistently with the outline given at the start of the section.

### Unit Group ( $A$ )

*/\* Input: The enveloping algebra,  $A$ , of a set  $X \subset \mathbb{M}(d, F)$  \*/*

*/\* Output: A generating set for  $U(A)$ , the group of units of  $A$  \*/*

- Step 1.** (a) Determine a composition series  $V_0 < V_1 < \dots < V_s$  for the natural  $A$ -module  $V = F^d$ , together with a change-of-basis matrix  $C$ .
- (b) Replace  $A$  with the algebra  $\text{Env}(CXC^{-1})$  so that elements of  $A$  have matrix as in Equation (1).
- (c) For  $1 \leq i \leq s$ , construct  $A_i$ , the algebra induced by  $A$  on  $V_i/V_{i-1}$ .
- (d) Define  $\varphi: A \rightarrow A_1 \oplus \dots \oplus A_s$ , sending  $a \mapsto (a_1, \dots, a_s)$ ; then  $B := \varphi(A)$  is a semisimple subalgebra of  $A_1 \oplus \dots \oplus A_s$ .
- (e) Construct a basis for  $A$ , and use it to obtain a matrix  $\mathcal{A}$  as in Equation (3).
- (f) Obtain the equivalence classes  $\Gamma_1, \dots, \Gamma_t$  of indices, wherein  $i \sim j$  if and only if  $V_i/V_{i-1}$  and  $V_j/V_{j-1}$  are isomorphic  $A$ -modules. Let the common dimension of the modules indexed by elements of  $\Gamma_i$  be  $e_i$ , and let  $\gamma_i$  be a representative of  $\Gamma_i$ . Also find the conjugating matrices  $c_\gamma$  for  $\gamma \in \Gamma_i$  as in Equation (2).

**Step 2.** Initialise  $Y := \emptyset$ . For  $1 \leq i \leq t$  proceed as follows:

- (a) Construct  $F_i$ , the centralising field of  $A_{\gamma_i}$  in  $\mathbb{M}(e_i, F)$  and set  $d_i := e_i/[F_i: F]$ .
- (b) Write down a generating set for the subgroup of  $\text{GL}(e_i, F)$  isomorphic to  $\text{GL}(d_i, F_i)$  contained within  $A_{\gamma_i}$ .
- (c) To each generator  $g$  in (b), assign  $y_g \in U(B)$  as follows. Initialise  $y_g := (1_{A_1}, \dots, 1_{A_s})$ . For  $\gamma \in \Gamma_i$ , insert  $c_\gamma g c_\gamma^{-1}$  in coordinate  $\gamma$  of  $y_g$ . Add  $y_g$  to  $Y$ .

**Step 3.** Construct  $S := \{\varphi^{-1}(y) : y \in Y\}$  using Lemma 2.1(i).

**Step 4.** Construct a generating set  $T$  for  $\ker(\varphi_U)$  using Lemma 2.1(iii).

**Step 5.** Return  $C^{-1}(S \cup T)C$ .

**Remark 2.2** The generating sets for the general linear group over finite fields, used in Step 2(b), are well known. Rónyai [13] points out that, from a complexity viewpoint, the problem of constructing these sets is equivalent to constructing a primitive element of the multiplicative group  $F^*$ , and that there is no known polynomial time algorithm to solve the latter problem. We assume that  $F$  has a known primitive element, which is the case in practice.

**Theorem 2.3** *The procedure **Unit Group** is a Las Vegas algorithm which, given an arbitrary matrix algebra  $A = \text{EnV}(X) \leq \mathbb{M}(d, F)$ , constructs a generating set for  $U(A)$ , the group of units of  $A$ . The algorithm uses  $O(td^6)$  operations in  $F$ , where  $t$  is the number of summands in the Wedderburn decomposition of  $A/J(A)$ .*

**Proof.** The correctness of the algorithm is clear from Lemma 2.1 and the discussion preceding it. It remains to examine the complexity of each step of the algorithm.

Variations of the MEATAXE algorithm are used in the following places: in Step 1(a) to find a composition series of  $V$ ; in Step 1(f) to determine isomorphisms between the modules  $V_i/V_{i-1}$ ; and in Step 2(a) to find the centralising fields  $F_i$ . These tasks are carried out using Las Vegas algorithms that use  $O(d^5)$  field operations; see [9, 11] for a detailed description.

The other steps are routine, typically involving linear algebra in the row space  $F^{d^2}$ . The basis in Step 1(e) is obtained using a standard transitive closure algorithm.  $\square$

**Remark 2.4** The Jacobson radical algorithm first constructs the composition series for the natural module  $V$  of the algebra  $A$  as described in Step (1), and then uses Lemma 2.1(ii) to construct generators for  $J(A)$ . Theorem 2.3 implies that this Las Vegas algorithm has the same complexity as **Unit Group**.

**Remark 2.5** If  $V$  has a composition factor upon which  $A$  acts as 0, then  $A$  has no units.

**Remark 2.6** Other algorithms having similar complexity could be employed to obtain the Wedderburn decomposition: see, for example, [5, 7, 10]. However, we found our approach, exploiting the composition series of the natural  $A$ -module, both more straight-forward, and simpler to implement.

The complexity of the unit group algorithm is determined by the cost of the techniques outlined in Lemma 2.1. It may be that an alternative approach which avoids such extensive use of linear algebra has better complexity.

### 3 Bases for two algebras

We sketch  $O(d^6)$  algorithms to construct the following subalgebras of  $\mathbb{M}(d, F)$ :

- (A) The algebra stabilising every subspace of a given lattice of subspaces of  $V = F^d$ .
- (B) The algebra centralising each matrix in a given set  $X \subset \mathbb{M}(d, F)$ .

Both algebras are crucial to the algorithm in Section 4. In each case we can readily construct an  $F$ -basis for the algebra. Denote by  $y_{ij}$  the indeterminate entries of an arbitrary matrix  $y \in \mathbb{M}(d, F)$ . In (A), for a subspace  $W$  of the lattice, the condition  $Wy \subseteq W$  is equivalent to solving a system of linear equations in the unknowns  $y_{ij}$ . A similar system arises in (B) from the condition  $xy = yx$  for  $x \in X$ . Thus in each case, a basis for  $A$  is obtained as the solution space of a system of equations in  $d^2$  unknowns over  $F$ .

### 4 The group preserving a system of forms

Let  $F$  be a finite field of size  $q$ , let  $d$  be a positive integer, and let  $V$  a vector space of dimension  $d$  over  $F$ . In this section we present an algorithm to solve the following problem:

Determine the subgroup,  $I$ , of  $\text{GL}(d, F)$  that preserves each form in a system  $\Sigma$  of bilinear or sesquilinear forms on  $V$ .

The basic approach is to construct a subalgebra,  $A$ , of  $\mathbb{M}(d, F)$  that contains  $I$  as a subset. Thus  $H := U(A)$  is an overgroup of  $I$ . As one component of the construction of  $A$ , we obtain a lattice of subspaces stabilised by both  $I$  and  $H$ . We next refine  $H$  so that the group it induces on each subspace of the lattice preserves the restriction of every form in  $\Sigma$  to that subspace. Finally, we construct  $I$  as a subgroup of  $H$ . A detailed description is given in Section 4.4.

## 4.1 Bilinear and sesquilinear forms

In this section we summarise the basic notions concerning classical forms on a vector space; see [16] for a comprehensive treatment of this subject.

Let  $(\cdot, \cdot)$  denote an alternating, symmetric or hermitian form on  $V$ ; we refer to these three possibilities collectively as *classical forms* on  $V$ . Each subspace  $U$  of  $V$  has an associated subspace  $U^\perp = \{v \in V : (u, v) = 0 \text{ for all } u \in U\}$  called the *perpendicular space of  $U$  relative to  $(\cdot, \cdot)$* . The *radical* of the form is the subspace  $V^\perp$ . The form is *degenerate* if its radical is nontrivial, and *nondegenerate* otherwise.

A classical form is *preserved* by  $g \in \text{GL}(V)$  if  $(v^g, w^g) = (v, w)$  for all  $v, w \in V$ . Each type of classical form has an associated field automorphism  $\zeta \mapsto \bar{\zeta}$ : if the form is alternating or symmetric then  $\bar{\zeta} = \zeta$ ; if it is hermitian then  $\bar{\zeta} = \zeta^{\sqrt{q}}$ . We extend this automorphism to  $V$  and to  $\mathbb{M}(d, F)$  in the obvious way.

Fix a basis  $v_1, \dots, v_d$  of  $V$ . It is convenient to identify a form with its associated matrix  $M = [[(v_i, v_j)]]$  where  $1 \leq i \leq d$  and  $1 \leq j \leq s$ . Under this identification, the value of  $(u, w)$  is  $uM\bar{w}^{\text{tr}}$ , and the radical of the form is the nullspace of  $M$ . Furthermore  $g \in \text{GL}(d, q)$  preserves  $M$  if  $(vg)M\bar{w}^g{}^{\text{tr}} = vM\bar{w}^{\text{tr}}$  for all  $v, w \in V$ . Thus the *isometry group* of  $M$  is defined as follows:

$$\text{Isom}(M) = \{ g \in \text{GL}(d, q) : gM\bar{g}^{\text{tr}} = M \}. \quad (5)$$

Let  $\Sigma = \{M_\omega : \omega \in \Omega\}$  be a system of (matrices representing) classical forms on  $V$ , where  $\Omega$  is a finite set. Define

$$\text{Isom}(\Sigma) = \bigcap_{\omega \in \Omega} \text{Isom}(M_\omega), \quad (6)$$

the subgroup of  $\text{GL}(d, q)$  preserving all forms in  $\Sigma$ . The next result, a direct extension of [2, Theorem 2.5], gives a useful description of  $\text{Isom}(\Sigma)$  if  $\Sigma$  contains a nondegenerate form.

**Lemma 4.1** *Let  $\Sigma$  be a system of classical forms, and let  $\Sigma_b$  and  $\Sigma_s$  denote the partition of  $\Sigma$  into bilinear and sesquilinear forms. Assume that there exist nonsingular  $M_b \in \Sigma_b$  and  $M_s \in \Sigma_s$ , and set*

$$X := \{NM_b^{-1} : N \in \Sigma_b\} \cup \{NM_s^{-1} : N \in \Sigma_s\}.$$

*Then  $\text{Isom}(\Sigma)$  is contained in the centraliser of  $X$  in  $\mathbb{M}(d, F)$ .*

## 4.2 Building an invariant lattice

The following procedure takes as input  $\Sigma$ , a system of classical forms on a vector space  $V$ , and constructs a lattice of subspaces of  $V$ , each of which is stabilised by  $\text{Isom}(\Sigma)$ . If  $W$  is a

subspace of  $V$  and  $M_\omega \in \Sigma$ , then  $W^\perp_\omega$  denotes the perpendicular space of  $W$  relative to  $M_\omega$ .

### Invariant Lattice ( $\Sigma$ )

*/\* Input: A system  $\Sigma = \{M_\omega : \omega \in \Omega\}$  of classical forms \*/*

*/\* Output: A lattice of subspaces of  $V$  stabilised by  $\text{Isom}(\Sigma)$  \*/*

**Step 0.** Initialise  $\mathcal{L} := \emptyset$ .

**Step 1.** For  $\omega \in \Omega$ , compute the nullspace,  $R_\omega$ , of  $M_\omega$ ; if  $R_\omega$  is a proper subspace, add it to  $\mathcal{L}$ .

**Step 2.** While there exists  $W \in \mathcal{L}$  and  $\omega \in \Omega$  such that  $W^\perp_\omega$  is a proper subspace of  $V$  not contained in  $\mathcal{L}$ , add  $W^\perp_\omega$  to  $\mathcal{L}$ .

**Step 3.** Return  $\mathcal{L}$ .

**Lemma 4.2** *The lattice output by Invariant Lattice consists of subspaces stabilised by  $\text{Isom}(\Sigma)$ .*

**Proof.** Let  $M_\omega \in \Sigma$  and let  $g \in \text{Isom}(\Sigma)$ . Then  $g \in \text{Isom}(M_\omega)$ , and so  $g$  preserves  $R_\omega$ . Hence  $\text{Isom}(\Sigma)$  stabilises each subspace in the initial lattice  $\mathcal{L}$  set up in Step 1.

It suffices to show that any new space added in Step 2 is stabilised by  $\text{Isom}(\Sigma)$ . Let  $W$  be any  $\text{Isom}(\Sigma)$ -invariant subspace, and let  $v \in W^\perp_\omega$  for some  $\omega \in \Omega$ . Then for all  $g \in \text{Isom}(\Sigma)$  and  $w \in W$  we have

$$wM_\omega \bar{v} \bar{g}^{\text{tr}} = wM_\omega \bar{g}^{\text{tr}} \bar{v}^{\text{tr}} = wg^{-1}M_\omega \bar{v}^{\text{tr}} = w'M_\omega \bar{v}^{\text{tr}}$$

for some  $w' \in W$ . Since  $w'M_\omega \bar{v}^{\text{tr}} = 0$ , it follows that  $W^\perp_\omega$  is stabilised by  $\text{Isom}(\Sigma)$ .  $\square$

## 4.3 Computing orbits on forms

The following procedure takes as input a classical form  $M$  on  $V$  and  $G \leq \text{GL}(V)$ , and uses permutation group techniques to construct generators for  $G \cap \text{Isom}(M)$ .

### Orbit Stabiliser ( $G, M$ )

*/\* Input: A matrix group  $G$  and a classical form  $M$  \*/*

*/\* Output: Generators for  $G \cap \text{Isom}(M)$  \*/*

**Step 1.** Compute the orbit  $\Delta := \{\bar{g}^{\text{tr}} M g : g \in G\}$ .

**Step 2.** Compute the subgroup  $G^*$  of  $\text{Sym}(\Delta)$  induced by  $G$  on  $\Delta$ , and construct a group epimorphism  $\Psi: G \rightarrow G^*$ .

**Step 3.** Construct a generating set  $X^*$  for the stabiliser in  $G^*$  of the point  $M$  of  $\Delta$ .

**Step 4.** Construct a preimage,  $X$ , of  $X^*$  under  $\Psi$ , and a generating set  $Y$  for the kernel of  $\Psi$ .

**Step 5.** Return the set  $\{\bar{z}^{\text{tr}} : z \in X \cup Y\}$ .

It is clear from its construction that the set returned by this procedure generates  $G \cap \text{Isom}(M)$ . The computation of the point-stabiliser within  $G^*$  in Step 3 is carried out using standard permutation group machinery; we refer to [14, Chapter 5] for a comprehensive treatment of these methods. The construction of  $X$  and  $Y$  in Step 4 is discussed in Section 5.

## 4.4 The isometry group algorithm

Let  $\Sigma = \{M_\omega : \omega \in \Omega\}$  be a system of classical forms. We now present a recursive procedure to construct generators for the isometry group  $\text{Isom}(\Sigma)$ .

### Isometry Group ( $\Sigma$ )

*/\* Input: A system  $\Sigma = \{M_\omega : \omega \in \Omega\}$  of classical forms on  $V = F^d$  \*/*

*/\* Output: A generating set for  $\text{Isom}(\Sigma)$  \*/*

**Step 1.** Let  $\Sigma_b$  and  $\Sigma_s$  denote the partition of  $\Sigma$  into bilinear and sesquilinear forms. If  $\Sigma_b$  contains a nonsingular matrix  $M_b$ , then set  $X_b := \{NM_b^{-1} : N \in \Sigma_b\}$ ; otherwise set  $X_b := \{I_d\}$ . Construct an analogous set  $X_s$  using  $\Sigma_s$ . As in Section 3, construct a basis for the algebra  $A_1$  centralising  $X := X_b \cup X_s$ .

**Step 2.** Use the algorithm in Section 4.2 to construct a lattice  $\mathfrak{L}$  stabilised by  $\text{Isom}(\Sigma)$ . As in Section 3, construct a basis for the algebra  $A_2$  stabilising  $\mathfrak{L}$ .

**Step 3.** Set  $A := A_1 \cap A_2$  and use the algorithm in Section 2.2 to construct  $H := U(A)$ .

**Step 4.** For each subspace  $W$  in  $\mathfrak{L}$ , proceed as follows:

- (a) Compute the subgroup  $H_W$  induced by  $H$  on  $W$  and a map  $\Psi_W : H \rightarrow H_W$ .
- (b) Compute the system  $\Sigma_W$  obtained by restricting the forms in  $\Sigma$  to  $W$ .
- (c) Recursively compute  $I_W := \text{Isom}(\Sigma_W)$ , a subgroup of  $\text{GL}(W)$ .
- (d) Construct generators for the intersection  $J_W := H_W \cap I_W$ .
- (e) Replace  $H$  by  $\Psi_W^{-1}(J_W)$ .

**Step 5.** Use the algorithm in Section 4.3 to construct the subgroup  $I := \text{Isom}(\Sigma)$  within  $H$ .

**Step 6.** Return  $I$ .

**Commentary.** Steps 1, 2 and 3 construct an overgroup  $H$  of the desired group  $I = \text{Isom}(\Sigma)$ . Lemmas 4.1 and 4.2 imply that  $I$  is contained in the algebras  $A_1$  and  $A_2$  respectively. Thus  $I \leq U(A)$  for the algebra  $A$  in Step 3. The intersection of the two algebras in Step 3 can be constructed readily using linear algebra.

Step 4 refines  $H$  so that the group it induces on each subspace of  $\mathcal{L}$  preserves the restriction of the forms in  $\Sigma$  to this subspace. In Section 5, we consider alternatives to the recursive call in (c) and discuss the construction of  $\Psi_W^{-1}(J_W)$  in (e). Finally, Step 5 constructs  $I$  as the subgroup of  $H$  which preserves each form in  $\Sigma$ . The success of this step rests on our ability to compute the orbit of  $H$  on a form  $M \in \Sigma$ , which is determined by the magnitude of the index of  $H \cap \text{Isom}(M)$  in  $H$ . Again, an alternative to this approach is discussed in Section 5.

## 4.5 Constructing intersections

In [2, Section 4.2] an algorithm is presented to descend from the full isometry group of a pair of forms of the same type, one of which is nondegenerate, to the intersection of the various pairs of classical groups that preserve those forms. The same procedure applies to the present setting. Thus, given a collection of groups  $\{G_\omega : \omega \in \Omega\}$ , where  $G_\omega$  preserves an alternating, symmetric or hermitian form (degenerate or nondegenerate), *and where this form is unique up to scalar multiple*, one can first obtain the corresponding system  $\Sigma = \{M_\omega : \omega \in \Omega\}$ , compute  $\text{Isom}(\Sigma)$  using the algorithm in Section 4.4, and then refine this group to obtain the intersection  $\bigcap_{\omega \in \Omega} G_\omega$ .

# 5 Implementation and performance

We implemented the algorithms presented in Section 2 and Section 4.4 in MAGMA [1]; they are publicly available. We now comment on aspects of these implementations and their performance.

## 5.1 The unit group algorithm

The performance of the unit group algorithm is limited primarily by the dimension of the systems of equations that must be solved in order to compute preimages under the epimorphism

$\varphi_U: U(A) \rightarrow U(B)$ . These systems have dimension roughly  $d^2$ , where  $d$  is the dimension of the input matrix generators. Linear systems having about 10,000 unknowns over “moderate” fields can be solved; thus we expect the unit group algorithm to be effective in dimensions up to about 100 over such fields.

**Performance.** The computations reported in Table 1 were carried out using MAGMA V2.13 on a Pentium IV 2.4 GHz processor. The input to the algorithm is the centralising algebra of a random invertible matrix in  $\mathbb{M}(d, q)$  (thus the algorithm returns the subgroup of  $GL(d, q)$  that centralises this matrix). In the column entitled “Time”, we list the CPU time in seconds taken to construct the group of units of this algebra. The time is averaged over five random selections.

Table 1: Performance of unit group algorithm

Input	Time
$\mathbb{M}(5, 5^{20})$	0.04
$\mathbb{M}(10, 5^2)$	0.05
$\mathbb{M}(20, 5^5)$	0.6
$\mathbb{M}(30, 5^5)$	3.8
$\mathbb{M}(40, 5^{10})$	20.9
$\mathbb{M}(50, 5^{10})$	54.0
$\mathbb{M}(60, 5^{10})$	177.2
$\mathbb{M}(80, 5^{10})$	235.7

## 5.2 The Jacobson radical algorithm

The performance of the Jacobson radical algorithm is limited primarily by the dimension of the systems of equations that must be solved in order to compute  $\ker \varphi$ . Hence the commentary from Section 5.1 applies.

**Performance.** The computations reported in Table 2 were carried out using MAGMA V2.13 on a Pentium IV 2.4 GHz processor.

Each algebra  $\text{Env}(X)$  is contained in  $\mathbb{M}(d, 5^{10})$ ; its semisimple quotient is the direct product of  $d/10$  copies of  $\mathbb{M}(10, 5^{10})$ ; and  $X$  has a single nilpotent matrix having  $d/10$  non-zero entries, each in the  $(k, k + 1)$  position, where  $k \equiv 0 \pmod{10}$ .

In the column entitled “New” and “CIW”, we list the CPU time in seconds taken to construct the Jacobson radical for  $\text{Env}(X)$  using respectively our algorithm and the general-purpose algorithm of [3].

Table 2: Performance of Jacobson radical algorithm

Input	New	CIW
$\mathbb{M}(20, 5^{10})$	0.2	1.0
$\mathbb{M}(30, 5^{10})$	0.5	3.7
$\mathbb{M}(40, 5^{10})$	1.4	8.8
$\mathbb{M}(50, 5^{10})$	3.6	18.6
$\mathbb{M}(60, 5^{10})$	8.6	34.4
$\mathbb{M}(70, 5^{10})$	16.4	60.0
$\mathbb{M}(80, 5^{10})$	29.5	115.6
$\mathbb{M}(90, 5^{10})$	57.1	195.9

### 5.3 The isometry group algorithm

We discuss various aspects of the implementation; the steps refer to the algorithm of Section 4.4.

**Step 2: Building the invariant lattice.** If  $\Sigma$  contains just two forms, then the lattice usually has moderate size, typically containing fewer than 20 subspaces. Otherwise the lattice is much larger, in which case  $\text{Isom}(\Sigma)$  usually has small order, often containing only scalar matrices. We limit the size of the lattice to 100 subspaces, since adding more subspaces does not generally refine the group preserving the lattice.

If  $\Sigma$  contains only nondegenerate forms, then the invariant lattice is empty and the algebra  $A$  in Step 3 is just the algebra  $A_1$  centralising the set  $X$  obtained in Step 1. In particular, if  $\Sigma$  contains precisely two nondegenerate forms, one bilinear and the other hermitian, then  $A$  is the full matrix algebra. In such cases our method is not applicable.

**Step 4: Recursion.** Following the recursive call in (c) to find generators for the group  $I_W = \text{Isom}(\Sigma_W)$ , the algorithm proceeds in (d) to compute the intersection  $J_W = H_W \cap I_W$ , using the standard back-track algorithm. To improve its chance of success, we ensure that the groups  $H_W$  and  $I_W$  are as small as possible. In particular, we process the subspaces  $W \in \mathfrak{L}$  in order of increasing dimension.

The back-track can be refined in our context, taking as input the group  $H_W$  and a membership test for  $I_W$  (of course  $g \in \text{GL}(W)$  is in  $I_W$  if and only if it preserves all of the forms in  $\Sigma_W$ ). Hence we could avoid the recursive call, and instead select those elements of  $H_W$  that lie in  $I_W$ . Another alternative is to find the subgroup of  $H_W$  preserving the forms in  $\Sigma_W$  via the permutation action of  $H_W$  on those forms, as in Step 5. We comment below on its practical limitations.

Our experience indicates that the approach taken in the paper – namely using a recursive call to compute  $\text{Isom}(\Sigma_W)$  – is the most effective, but we have encountered examples where the alternatives complete but the basic algorithm does not.

**Steps 4 and 5: Constructing kernels and preimages.** Suppose that we have a group homomorphism  $\Psi: L \rightarrow S$  from a “large” group  $L$  to a “small” group  $S$ . If  $X = \{x_1, \dots, x_m\}$  is the given generating set for  $L$ , then we assume that  $\Psi$  is given by specifying the image,  $\overline{X} = \{\overline{x_1}, \dots, \overline{x_m}\}$ , of  $X$  under  $\Psi$ . We also assume that we can efficiently compute the  $\Psi$ -image in  $S$  of an arbitrary element of  $L$ . Since  $S$  is small, we may also assume that we can compute effectively within  $S$  using standard machinery.

In our context, we require efficient algorithms to compute the preimage of an arbitrary element of  $S$ , and to construct a generating set for the kernel of  $\Psi$ . Such are needed in the main algorithm in Step 4 (e) to construct the preimage  $\Psi_W^{-1}(J_W)$ , and also in Step 5, where the **Orbit Stabiliser** algorithm of Section 4.3 is used repeatedly.

Preimages under  $\Psi$  are handled readily as follows. Write the given  $s \in S$  as a word in  $\overline{X}$ , say  $s = w_s(\overline{x_1}, \dots, \overline{x_m})$ . (This is the essential algorithmic task that we assume can be carried out within  $S$ .) A suitable preimage is then obtained by evaluating  $w_s(x_1, \dots, x_m)$  in  $L$ .

The kernel  $K$  of  $\Psi$  may be constructed using, for example, the techniques of [12]. If  $L$  is a permutation or matrix group, then these assume that a *base and strong generating set* is available for  $L$ . Although such can be computed using the Random Schreier algorithm [14, Chapter 4], it is sometimes very expensive, particularly for matrix groups.

Since  $L$  is often a large matrix group in our setting, we instead use a Monte Carlo algorithm to construct generators for  $K$ . Choose random  $l \in L$ , and evaluate  $s := \Psi(l) \in S$ . Write  $s = w_s(\overline{x_1}, \dots, \overline{x_m})$  as a word in  $\overline{X}$ . Then  $l \cdot w_s(x_1, \dots, x_m)^{-1} \in K$ . Repeating this construction for sufficiently many random elements of  $L$ , with *high probability* we obtain a generating set for  $K$ . Our implementation can use either the deterministic or randomised algorithm to construct  $K$ .

**Step 5: Orbits on forms.** The applicability of this algorithm depends on the size of the orbit of  $H$  on a form  $M \in \Sigma$ , or equivalently on the index of  $\text{Isom}(M) \cap H$  in  $H$ . In our implementation, we construct orbits of size at most  $10^6$ ; if an orbit is larger, we return  $H$  and report that it is an

overgroup of the desired group.

**Performance.** In Table 3 we report on a number of computations of isometry groups: the input is a system of  $r$  bilinear or sesquilinear forms contained in  $\mathbb{M}(d, q)$ . Of the  $r$  forms,  $s$  are non-degenerate. We list the CPU time in seconds to construct the intersection.

Table 3: Performance of isometry group algorithm

Input	$r/s$	Time
$\mathbb{M}(4, 5^6)$	3/3	0.8
$\mathbb{M}(6, 5^6)$	4/0	0.2
$\mathbb{M}(8, 5^6)$	3/2	0.2
$\mathbb{M}(9, 11)$	2/0	109.9
$\mathbb{M}(10, 5^6)$	3/2	0.2
$\mathbb{M}(11, 7)$	2/0	806.7
$\mathbb{M}(12, 5^6)$	3/0	2.4
$\mathbb{M}(14, 5^6)$	4/0	22.5
$\mathbb{M}(16, 5^6)$	3/0	3.8

## References

- [1] W. Bosma, J. Cannon and C. Playoust. The MAGMA algebra system I: The user language. *J. Symbolic Comp.* 24 (1997) 235–265.
- [2] Peter A. Brooksbank and E.A. O’Brien. On intersections of classical groups. Accepted to appear, *J. Group Theory*, 2008.
- [3] Arjeh M. Cohen, Gábor Ivanyos and David B. Wales. Finding the radical of an algebra of linear transformations. *J. Pure Appl. Algebra* 117/118 (1997), 177–193.
- [4] Charles W. Curtis and Irving Reiner. Representation theory of finite groups and associative algebras. Reprint of the 1962 original. AMS Chelsea Publishing, Providence, RI, 2006.
- [5] W. Eberly and M. Giesbrecht. Efficient decomposition of associative algebras over finite fields. *J. Symbolic Comput.* 29 (2000), no. 3, 441–458.

- [6] Bettina Eick, C.R. Leedham-Green and E.A. O'Brien. Constructing the automorphism group of a  $p$ -group. *Comm. Algebra*, 30, 2271-2295, 2002.
- [7] K. Friedl and L. Rónyai. Polynomial time solutions of some problems in computational algebra. pp. 153-162 in: *7th Ann. Symp. Theory of Comp.* Providence, RI, USA, 1985.
- [8] Daniel Goldstein and Robert M. Guralnick. Alternating forms and self-adjoint operators. *Journal of Algebra* 308 (2007), 330–349.
- [9] Derek F. Holt and Sarah Rees. Testing modules for irreducibility. *J. Austral. Math. Soc. Ser. A* 57 (1994), no. 1, 1–16.
- [10] Gábor Ivanyos. Fast randomized algorithms for the structure of matrix algebras over finite fields (extended abstract). *Proceedings of the 2000 International Symposium on Symbolic and Algebraic Computation (St. Andrews)*, 175–183 (electronic), ACM, New York, 2000.
- [11] Gábor Ivanyos and Klaus Lux. Treating the exceptional cases of the MeatAxe. *Experiment. Math.* 9 (2000), no. 3, 373–381.
- [12] Charles R. Leedham-Green, Cheryl E. Praeger and Leonard H. Soicher. Computing with group homomorphisms. *J. Symbolic Comput.* 12 (1991), no. 4-5, 527–532.
- [13] Lajos Rónyai. Computing the order of centralizers in linear groups. *Inform. and Comput.* 91 (1991), no. 2, 172–176.
- [14] Ákos Seress. *Permutation group algorithms*, volume 152 of *Cambridge Tracts in Mathematics*. Cambridge University Press, Cambridge, 2003.
- [15] Ruth Schwingel. Two matrix group algorithms with applications to computing the automorphism group of a finite  $p$ -group. PhD thesis, Queen Mary, University of London, 2000.
- [16] Donald E. Taylor. *The geometry of the classical groups*. Heldermann, Berlin, 1992.

*Peter A. Brooksbank*  
*Department of Mathematics*  
*Bucknell University*  
*Lewisburg, PA 17837*  
*United States*  
 pbrooksb@bucknell.edu

*E.A. O'Brien*  
*Department of Mathematics*  
*University of Auckland*  
*Private Bag 92019*  
*New Zealand*  
 obrien@math.auckland.ac.nz

Last revised August 2007