Computing matrix group decompositions with respect to a normal subgroup

Derek F. Holt, C.R. Leedham-Green,
E.A. O'Brien and Sarah Rees

Abstract

We describe an algorithm which can be used to investigate whether a matrix group defined over a finite field decomposes with respect to a normal subgroup, defined as the normal closure of a given set of matrices. The possible decompositions correspond to classes in Aschbacher's classification of subgroups of the general linear group.

1 Introduction

The purpose of this paper is to describe the algorithm Smash, which has been developed by the authors as a tool to aid the computer recognition of a group described by a generating set of matrices over a finite field. Given a set S of matrices in a group G, not all of which are scalar, Smash looks for certain kinds of decompositions of G and the underlying vector space V with respect to the normal subgroup $N = \langle S \rangle^G$, the normal closure of the subgroup generated by S. The algorithm has already been implemented in the GAP system, (see [13]), and the development of the algorithm has been strongly influenced by the performance of that implementation. The GAP code is available from the authors. An implementation will also be available within Magma [2].

This paper is divided into six sections, the first of which is introductory. Section 2 describes the theoretical basis for the algorithm, and Section 3 lists the main steps, while the main procedures are described in greater detail in Section 4. Section 5 addresses the issues of termination and complexity, and Section 6 gives performance statistics, and suggests possible improvements to the algorithm.

Aschbacher's theorem (in [1]) on the structure of subgroups of the general linear group $GL(d, q)$ is stated in more detail below. Essentially, it shows that if G is a subgroup of $GL(d, q)$, then, modulo scalars, G is either almost simple, or embeds in $GL(d, q')$ for some q' dividing q, or belongs to one of a number of classes of subgroups of $GL(d, q)$ which naturally give rise to a normal subgroup N of G. Two possible decompositions which involve a normal subgroup occur when G acts reducibly on the vector

The authors thank the School of Mathematical Sciences, Australian National University, for its hospitality while much of this work was carried out.

space V (that is, preserves a subspace of V), or irreducibly but not absolutely irreducibly (in which case G is isomorphic to a subgroup of $GL(d/e, q^e)$ for some $e > 1$, and is thus in Aschbacher's class of subgroups which embed isomorphically in $\Gamma L(d/e, q^e)$). These cases are disposed of using a variation of Parker's MEATAXE (see [11, 7]). Once these possibilities have been eliminated, SMASH seeks to recognise the remaining classes which involve a normal subgroup.

SMASH was initially developed as one component of an algorithm which can be used to decide whether or not a matrix group is primitive. That algorithm is described in detail in [6].

SMASH and the algorithm for primitivity testing have been written as part of a general project to produce fast algorithms to recognise matrix groups, inspired by the development of two Monte Carlo algorithms to recognise the special linear group by Neumann and Praeger, (see [10]) and by Leedham-Green ([8], and see also [9] for a description of a non-deterministic constructive algorithm). Both of these algorithms make use of the classification by Aschbacher mentioned above and further use of this for more general identification of a matrix group is proposed in [12]. Aschbacher's list has been the basis of all the algorithms developed in this project to date.

The classification provided by Aschbacher's theorem, reproduced here essentially from [10], is given below. Here, as throughout this paper, G is a group of $d \times d$ matrices over the finite field $F = GF(q)$, defined by a set of generating matrices, and V is the underlying d-dimensional vector space over $GF(q)$ on which G acts.

Theorem 1 (Aschbacher, [1])
Let G be a subgroup of $GL(d, q)$, and let Z denote its subgroup of scalar matrices, that is $Z = Z(GL(d, q)) \cap G$. Then one of the following is true:-

1. *G acts reducibly, that is, G preserves a proper subspace of V.*
2. *G acts imprimitively, that is, G preserves a decomposition of V as a direct sum $V_1 \oplus V_2 \oplus \cdots \oplus V_r$ of $r > 1$ subspaces of dimension s, which are permuted transitively by G, and so $G \subseteq GL(s, q) \wr S_r$.*
3. *G preserves a decomposition of V as a tensor product $U \otimes W$ of spaces of dimensions $r, s > 1$ over F. Then G is a central product of subgroups of $GL(r, q)$ and $GL(s, q)$. More precisely, $G/Z \subseteq PGL(r, q) \times PGL(s, q)$.*
4. *G preserves a decomposition of V as a symmetric tensor product $V_1 \otimes V_2 \otimes \cdots \otimes V_m$ of spaces all of dimension $r > 1$ over F, where $d = r^m$. The components of the product are permuted by G, and so G is an amalgamated wreath product of a subgroup of $GL(r, q)$ by a subgroup of S_m. More precisely, $G/Z \subseteq PGL(r, q) \wr S_m$.*
5. *G acts on V as a group of semilinear automorphisms of a d/e-dimensional space over the extension field $GF(q^e)$, for some $e > 1$, so G embeds in $\Gamma L(d/e, q^e)$.*
6. *Modulo Z, G is conjugate to a subgroup of $GL(d, q')$, for some proper subfield $GF(q')$ of $GF(q)$, that is, $G^g \subseteq GL(d, q').Z$, for some $g \in GL(d, q)$.*
7. *For some prime r, $d = r^m$, and G is contained in the normaliser of an r-group R, of order either r^{2m+1} or 2^{2m+2}. Either R is extraspecial (in the first case), or R is a 2-group of symplectic type, that is, a central product of an extraspecial 2-group with a cyclic group of order 4.*

8. *G/Z contains the derived subgroup of $PGO(d,q)$, $PGSp(d,q)$, $PGU(d,q)$ or $PGL(d,q)$, and G itself is a subgroup of $GO(d,q)Z$, $GSp(d,q)Z$, $GU(d,q)Z$ or $GL(d,q)Z$ respectively.*

9. *$T \subset G/Z \subseteq Aut(T)$, for some non-abelian simple $T = G_0/Z$, for some subgroup G_0 of G.*

The nature of the embedding in the fifth case needs to be described more precisely. In this case, V can be regarded as a vector space of dimension d/e over $GF(q^e)$ and, for each $g \in G$, there is an automorphism $g\alpha$ of $GF(q^e)$ fixing $GF(q)$, such that $(\lambda v)^g = \lambda^{g\alpha} v^g$ for all $v \in V$ and $\lambda \in GF(q^e)$. We denote the group of semilinear transformations of V of this form by $\Gamma L(d/e, q^e)$. We stress that this notation means the subgroup of the full semilinear group consisting of those maps for which the associated field automorphism α of $GF(q^e)$ fixes the field $F = GF(q)$. (So, for example $\Gamma L(2, 2^4)$ and $\Gamma L(2, 4^2)$ are nonisomorphic groups.) In this situation, we shall simply say that G embeds in $\Gamma L(d/e, q^e)$, although this is a little sloppy, because there are many different embeddings of $\Gamma L(d/e, q^e)$ in $GL(d, q)$. We shall also describe this case by saying simply that G is semilinear.

Under the assumption that G has already been shown to act absolutely irreducibly on the space V, and given a set S of matrices of G, not all of which are scalar, SMASH investigates whether G has one of the following decompositions with respect to the normal subgroup $N = \langle S \rangle^G$:-

(i) G acts imprimitively on V, with blocks $V_1, V_2, \ldots, V_r$, and N preserves each of the subspaces V_i.

(ii) G is a group of semilinear (but not linear) automorphisms in some dimension dividing d, over an extension field of $GF(q)$. Thus G embeds in $\Gamma L(d/e, q^e)$ but not in $GL(d/e, q^e)$, and $N \subseteq G \cap GL(d/e, q^e)$.

(iii) G preserves a tensor product decomposition $U \otimes W$ of V, and N acts as scalar matrices on U. Thus N preserves a decomposition of V as a direct sum of subspaces isomorphic to W, and fixed by N.

(iv) For some prime r, where $d = r^m$, G is the normaliser of an r-group, R, of order r^{2m+1}, or 2^{2m+2}. In this case, N is contained in RZ.

(v) G preserves a symmetric tensor product of m spaces each of dimension r, and is an amalgamated wreath product of a subgroup of $GL(r, q)$, and a subgroup of the symmetric group S_m, for some r, m with $d = r^m$. Then N preserves each of the m factors of the tensor product.

The investigations of the first four types of decompositions are conclusive, that is, if G decomposes in one of these ways with respect to N, SMASH will identify at least one such decomposition. However the investigation of the symmetric tensor product decomposition is at present a non-deterministic algorithm, and so the failure to find such a decomposition does not prove definitely its non-existence.

2 The theory behind Smash

Suppose that G acts absolutely irreducibly on the d-dimensional space V over $F = GF(q)$, and that N is a normal non-scalar subgroup of G. Then, by Clifford's theorem (see [3], or [5]), for some $t \geq 1$, V splits as a direct sum $W_1 \oplus W_2 \oplus \cdots \oplus W_t$ of irreducible FN-modules, all of the same dimension. For some $r, s' \geq 1$, with $rs' = t$, the W_i's partition into r sets containing s' pairwise isomorphic FN-modules each, and if $V_1, V_2, \ldots, V_r$ are each the sum of s' pairwise isomorphic W_i's, so that $V = V_1 \oplus V_2 \oplus \cdots \oplus V_r$, then G permutes the V_i's transitively.

If $r > 1$, then the subspaces V_i form the blocks of a non-trivial system of imprimitivity, and N preserves each V_i. Thus we have case (i) of the above list of decompositions.

If $r = 1$, then V decomposes as a direct sum of t irreducible, pairwise isomorphic FN-modules, $W_1 = W, W_2, \ldots, W_t$, each of dimension $d' = d/t$ over F. Either all of the W_i are absolutely irreducible as FN-modules or all are not. There is an integer $e \geq 1$ such that, for each i, the matrices describing the action of N on W_i can be written as $d'/e \times d'/e$ matrices over $GF(q^e)$, and W_i can then be regarded as an absolutely irreducible module for N of dimension d'/e over $GF(q^e)$. In particular W is absolutely irreducible as a $GF(q^e)N$-module. Now we apply Schur's lemma (see, for example, [5]), which states that any non-trivial endomorphism of an absolutely irreducible $GF(q^e)N$-module corresponds to multiplication by a scalar matrix over $GF(q^e)$. Hence we see that the endomorphism ring $\mathrm{Hom}_{FN}(W, V)$ is t-dimensional as a vector space over $GF(q^e)$, with $\{\theta_1, \theta_2, \ldots, \theta_t\}$ as a basis, where θ_1 is the identity map from W to itself, and θ_i is an isomorphism from W to W_i.

Suppose first that the W_i are not absolutely irreducible over $GF(q)$, that is, that $e > 1$. Then $N|_{W_i}$, the restriction of N to W_i, embeds in $GL(d'/e, q^e)$ for each i. With respect to this embedding, a basis for W as a d'/e-dimensional space over $GF(q^e)$ can be found by computing the centralising ring for W. Using Schur's lemma again, we see that this is isomorphic to the field $GF(q^e)$, or rather, the ring of $d'/e \times d'/e$ scalar matrices over that field. Applying each of the maps θ_i, for $i > 1$, we can extend this basis to a basis B for the whole of V, and thus find an embedding of N in $GL(d/e, q^e)$. We shall demonstrate that G is a subgroup of $\Gamma L(d/e, q^e)$ (not in $GL(d/e, q^e)$) by exhibiting an appropriate subgroup K of $GL(d/e, q^e)$ which is normalised by G.

Since $\{\theta_1, \theta_2, \ldots, \theta_t\}$ forms a basis for $\mathrm{Hom}_{FN}(W, V)$ as a vector space over $GF(q^e)$, every translate of W under G can be found as the image of W under a map of the form $\sum_{i=1}^{t} \Lambda_i \theta_i$, where Λ_i is a linear transformation of W corresponding to field multiplication by an element of $GF(q^e)$. Let L be the subgroup of $GL(d, q)$ which fixes all the translates of W by G. Then L is normalised by G. It is fairly easy to see that, with respect to the basis B, each element of L is made up of t copies of a single $d' \times d'$ matrix A along the diagonal, where A corresponds to an element of $GL(d'/e, q^e)$ embedded in $GL(d', q)$. Therefore L is isomorphic to $GL(d', q)$. Let K be the centraliser of N in L. Then K is also normalised by G, and is isomorphic to the group of $d'/e \times d'/e$ scalar matrices over $GF(q^e)$. This embeds in $GL(d, q)$ as a group of matrices, each with d/e identical $e \times e$ blocks along the diagonal. The normaliser of K in $GL(d, q)$ is $\Gamma L(d/e, q^e)$, so since G normalises K, G must be a subgroup of $\Gamma L(d/e, q^e)$. (The inclusion of G as a subgroup of $GL(d/e, q^e)$ is prohibited by the fact that G acts absolutely irreducibly on V.) Thus we have case (ii) of the list of decompositions.

From now on, we shall assume that the W_i are all absolutely irreducible over $GF(q)$.

We consider separately the cases $t > 1$ and $t = 1$.

Suppose first that $t > 1$. Choose a basis $B = \{b_{ij} : 1 \leq i \leq t, 1 \leq j \leq d'\}$ for V such that, for each i, $\{b_{ij} : 1 \leq j \leq d'\}$ is a basis for W_i, and for each i, j, $b_{ij} = b_{1j}\theta_i$. Then G preserves a decomposition of V as a tensor product $U \otimes W$, where W is an N-module isomorphic to each W_i, and N acts as scalars on the t-dimensional space U. We now describe how to find the tensor decomposition with respect to the basis B.

First, for any $g \in G$, the translate, Wg, of W by g is isomorphic to W as an FN-module, and we can find an FN-module automorphism θ_g of V such that $Wg = W\theta_g$. Since the maps θ_i form a basis for $\mathrm{Hom}_{FN}(W, V)$ we can find elements $x_{11}, x_{12}, \ldots, x_{1t}$ of F such that $\theta_g|_W = x_{11}\theta_1 + x_{12}\theta_2 + \ldots + x_{1t}\theta_t$. Now $g\theta_g^{-1}$ fixes W, and so there is an F-linear map ϕ on W such that $g|_W = \phi\theta_g|_W$. Where ϕ is represented by a $d' \times d'$ matrix $y = (y_{ij})$ over F with respect to the basis $\{b_{11}, b_{12}, \ldots, b_{1t}\}$ of W, we see that the first d' rows of the matrix representing g with respect to B consist of t blocks of size $d' \times d'$ from left to right, corresponding to the matrix y multiplied by each of $x_{11}, x_{12}, \ldots, x_{1t}$.

To complete the picture we need to consider the action of g on the translates $W_i = W\theta_i$ of W, for each $i > 1$. But $\theta_i g = g\theta_i^g$. So the rows of the matrix for g with respect to B which correspond to its action on W_i are the same as the rows of the action on W of $g\theta_i^g$ and hence of $\phi\theta_g\theta_i^g$. And since N is normal in G, and θ_i commutes with the action of N, θ_i^g commutes with the action of N. So $\theta' = \theta_g\theta_i^g$ is an FN-module homomorphism from W to V. Hence we can find $x_{i1}, x_{i2}, \ldots, x_{it}$ such that $\theta'|_W = x_{i1}\theta_1 + x_{i2}\theta_2 + \ldots + x_{it}\theta_t$, and so we see that rows $d'(i-1)+1$ to $d'i$ of the matrix representing g with respect to B consist of t blocks of size $d' \times d'$ from left to right, corresponding to the matrix y multiplied by each of $x_{i1}, x_{i2}, \ldots, x_{it}$. So the matrix for g is clearly the Kronecker product of the $t \times t$ matrix $x = (x_{ij})$ and the $d' \times d'$ matrix $y = (y_{ij})$. Thus we have case (iii) of the list of decompositions.

Now suppose that $t = 1$. The group N acts absolutely irreducibly, so its centre $Z(N)$ consists of scalar matrices, and hence is cyclic.

Suppose also that N is minimal with respect to $N/Z(N)$ being non-trivial. (The theory below depends on this assumption, although there are particular circumstances where a decomposition might be found, even if N did not satisfy this condition.) Then $N/Z(N)$ is a direct product $N_0 \times N_0 \times \cdots \times N_0$ of m copies of a simple group N_0, and N is a central product of m groups N_1, each isomorphic to an extension of $Z(N)$ by N_0.

First suppose that N_0 is cyclic, and therefore that $N/Z(N)$ is an elementary abelian r-group for some prime r. We claim that then N is extraspecial or of symplectic type (that is, a central product of an extraspecial 2-group with a cyclic group of order 4).

Since $N/Z(N)$ is non-trivial, N cannot be abelian. For x, y in N, we have $[x, y]^r = [x^r, y] = 1$ (since x^r is in $Z(N)$), so N' has exponent r. Since N' lies in $Z(N)$, which is cyclic, we must have $|N'| = r$.

First suppose that r is odd. Let M be the set of elements of N of order dividing r. Since N' is central, for any x, y in N we have $(xy)^r = x^r y^r [y, x]^{r(r-1)/2}$, and so, since N' has exponent r, M is a subgroup of N. But M is clearly characteristic in N, and hence normal, so by the minimality of N, either $M \subseteq Z(N)$, or $M = N$. For r odd, an r-group containing a unique subgroup of order r is cyclic, so N contains more than one subgroup of order r. Hence M is not cyclic, and so cannot lie in $Z(N)$. Therefore $N = M$, so $Z(N) = N'$ has order r, and N is extraspecial of exponent r.

Now suppose that $r = 2$. Then N' has order 2, and $N/Z(N)$ is an elementary abelian 2-group. This time let M be the set of elements of N of order dividing 4. Then for x, y in N, as above, $(xy)^4 = x^4y^4[y,x]^6$, so again M is a characteristic subgroup of N, and hence is normal in N. Again, by the minimality of N, either $M \subseteq Z(N)$ or $M = N$.

If $M \subseteq Z(N)$, then, suppose that x and y are non-central elements of N, with $xZ(N) \neq yZ(N)$, and with the order of x greater than or equal to the order of y. Then x and y are both outside M, so both have order greater than 4. Also, x^2 and y^2 are central (since $N/Z(N)$ has exponent 2). Therefore, since $Z(N)$ is cyclic, for some c, $y^2 = x^{2c}$, and $(yx^{-c})^4 = (y^2)^2(x^{-2c})^2[x^{-c},y]^6 = 1$. Thus yx^{-c} is in M, and hence in $Z(N)$, contradicting the fact that $xZ(N)$ and $yZ(N)$ are distinct non-trivial elements of the elementary abelian 2-group $N/Z(N)$. Therefore $M = N$.

Since N has exponent 4 and $Z(N)$ is cyclic, $Z(N)$ has order 2 or 4. Either $Z(N)$ has order 2, and N is extraspecial, or it has order 4, and N is of symplectic type. Thus we have case (iv) of the decomposition.

Finally suppose that N_0 is non-abelian simple. If $m = 1$, G is almost simple, and Smash fails to find a decomposition. If $m > 1$ we get case (v) of the list of decompositions. With respect to an appropriate basis, V can be decomposed as a tensor product $X_1 \otimes X_2 \otimes \cdots \otimes X_m$ of spaces on each of which N acts as N_1. Conjugation of N by G permutes the central factors of N and hence the factors of the tensor product. So $G/Z(G)$ is isomorphic to a wreath product of N_0 with a subgroup of S_m.

3 The main algorithm

We assume that G acts absolutely irreducibly on the d-dimensional vector space V over $F = GF(q)$. We are given S as a set of matrices in G, not all of which are scalar. We use $V_{\langle S\rangle}$ to mean V viewed as an $F\langle S\rangle$-module.

Note that the set S generates N as a normal subgroup, but not necessarily as a subgroup, that is, $N = \langle S\rangle^G$ but it is not necessarily true that $N = \langle S\rangle$. The algorithm deals with this by adding conjugates of elements of S to S as necessary.

The basic idea is first to find a breakdown of $V_{\langle S\rangle}$ as a direct sum of irreducible $F\langle S\rangle$-modules $W_1, W_2, \ldots, W_t$. Then, if $t > 1$, or $t = 1$ but $\langle S\rangle$ does not act absolutely irreducibly on W, we attempt to recognise one of the three decompositions where G acts imprimitively, or preserves a tensor product, or is a non-linear subgroup of a semilinear group, using the functions MinBlocks, TensorProduct, and SemiLinear. If none of those situations is found, it must be because $\langle S\rangle \neq \langle S\rangle^G$ and hence the modules W_i are not in fact irreducible $F\langle S\rangle^G$-modules, so S is enlarged through the addition of conjugates, and the algorithm is restarted with the new enlarged S. This iteration continues until either a decomposition is found (in which case Smash returns that decomposition) or $\langle S\rangle$ is found to act absolutely irreducibly on V. In the second case, we apply the tests ExtraSpecial and SymTensorProduct, in an attempt to identify G either as a normaliser of a group of prime power order or as a group preserving a symmetric tensor product. If a decomposition is found, it is returned by Smash. Otherwise, Smash returns false.

The tests called by Smash and mentioned above are described in more detail in Section 4.

The algorithm proceeds as follows.

Step 1. Find a random irreducible $F\langle S\rangle$-submodule W of $V_{\langle S\rangle}$. If $W \neq V$ is found, go to Step 2. Otherwise, that is, if $V_{\langle S\rangle}$ is irreducible, go to Step 8.

Step 2. Try to express $V_{\langle S\rangle}$ as a direct sum of irreducible $F\langle S\rangle$-modules which are translates of W. First check if the translates of W are direct summands, then check if they are $F\langle S\rangle$-modules, then finally check if they are irreducible. If any one of these checks fails, a conjugate of an element of S is identified which does not fix W. Add this to S, and restart. Otherwise, $V_{\langle S\rangle}$ is a direct sum $W_1 \oplus W_2 \oplus \cdots \oplus W_t$ of irreducible submodules W_i, where each W_i is a translate of $W = W_1$. Go to Step 3.

Step 3. Use MinBlocks to try and find a system of imprimitivity with W as a subspace of one of the blocks. If MinBlocks returns false, go to Step 4, otherwise return the system of imprimitivity found by MinBlocks.

Step 4. Use IsomMod to test each pair (W_1, W_i), where $i > 1$, for isomorphism as $F\langle S\rangle$-modules. If all pairs are isomorphic, find a basis $B = \{b_{ij} : 1 \leq i \leq t, 1 \leq j \leq d'\}$ of V, such that, for each i, $\{b_{ij} : 1 \leq j \leq d'\}$ is a basis for W_i, and the isomorphism from W_1 to W_i maps each b_{1j} to b_{ij}, and go to Step 5. Otherwise, add to S a random conjugate which does not fix W, and restart.

Step 5. Test W for absolute irreducibility as an $F\langle S\rangle$-module. If W is absolutely irreducible, go to Step 6, otherwise go to Step 7.

Step 6. Use the TensorProduct test to try and write each generating matrix of G as a tensor product of matrices of dimensions t and d' with respect to the basis B found in Step 4. If this fails, add to S a random conjugate which does not fix W, and restart, otherwise return the tensor decomposition found by TensorProduct.

Step 7. Let c be a $d' \times d'$ matrix generating the field of centralising elements of the action of $\langle S\rangle$ on W. Construct the $d \times d$ matrix C which acts as c on each of the isomorphic modules W_i. Use the SemiLinear test to see if G embeds in $\Gamma L(d/e, q^e)$ with S in $GL(d/e, q^e)$ and C as a $d/e \times d/e$ scalar matrix over $GF(q^e)$. If the SemiLinear test returns false add to S a conjugate which does not commute with C, and restart. Otherwise return the decomposition found by SemiLinear.

Step 8. At this stage $V_{\langle S\rangle}$ must be irreducible. Test $V_{\langle S\rangle}$ for absolute irreducibility. If $V_{\langle S\rangle}$ is absolutely irreducible, go to Step 10. Otherwise go to Step 9.

Step 9. Let C be a $d \times d$ matrix generating the field of centralising elements of the action of $\langle S\rangle$ on V. Use the SemiLinear test to see if G embeds in $\Gamma L(d/e, q^e)$ with S in $GL(d/e, q^e)$ and C as a $d/e \times d/e$ scalar matrix over $GF(q^e)$. If the SemiLinear test returns false add to S a conjugate which does not commute with C, and go back to Step 8. Otherwise return the decomposition found by SemiLinear.

Step 10. If $d = r^m$ for a prime r, use the ExtraSpecial test to see if G normalises an r-group, and if so return a set of generators for the r-group and the action of G on those. Otherwise go to Step 11.

Step 11. Use the SymTensorProduct test to attempt to find a decomposition of G as an amalgamated wreath product of a subgroup of $GL(r,q)$ by a subgroup of the symmetric group S_m, where $d = r^m$. Return such a decomposition, if it can be found, otherwise return false.

4 The procedures called by Smash

The tests for irreducibility and absolute irreducibility of modules, and also the test IsomMod for isomorphism between two modules are all described in [7]. The test for absolute irreducibility computes a centralising matrix as required, and the test for isomorphism returns appropriate bases. Random irreducible submodules of $V_{\langle S\rangle}$ are generated, where necessary, using a modification of the test for irreducibility.

The procedure MinBlocks, which searches conclusively for a system of imprimitivity containing a specified subspace as a subspace of a block, is described in [6]. It remains to describe the procedures SemiLinear, TensorProduct, ExtraSpecial and SymTensorProduct.

4.1 The test SemiLinear

As input for SemiLinear we have G, a set S of elements of G, a matrix C of G, and an integer e. The subset S of G is known to embed in $GL(d/e, q^e)$, for $e > 1$, and the $d \times d$ matrix C is known to act as multiplication by a scalar λ (a field generator of $GF(q^e)$) for that embedding. The matrix C is, of course, central in $GL(d/e, q^e)$. Then G acts as a semilinear group of automorphisms on the d/e-dimensional space if and only if, for each generator g of G, there is an integer $i = i(g)$ such that $Cg = gC^{q^i}$, that is, g corresponds to the field automorphism $\lambda \to \lambda^{q^i}$. In that case, we have a map from G to the (cyclic) group $\mathrm{Aut}(GF(q^e))$, and C centralises the kernel of this map, which thus lies in $GL(d, q^e)$. We test this as follows. First, if possible, we find $i = i(g)$ such that $wCg = wgC^{q^i}$ for a single vector w of the d-dimensional space (in fact the first vector of the standard basis) and then we check that $vCg = vgC^{q^i}$ for all other vectors v in the basis. The test returns false if no such $i(g)$ is found, for some generator g.

4.2 The test TensorProduct

As input for TensorProduct we have G, V, and the basis $B = \{b_{ij} : 1 \leq i \leq d_1, 1 \leq j \leq d_2\}$. We seek to decompose the action of G on V as a tensor product of spaces of dimensions d_1 and d_2, with respect to B. We simply run through each of the generating matrices g of G in turn, and try to express $g = (g_{ij})$ as a tensor product with respect to that basis. To do this we need to find a $d_1 \times d_1$ matrix $x = (x_{ij})$ and a $d_2 \times d_2$ matrix $y = (y_{ij})$ with $g = x \otimes y$. In this case,

$$g_{(i_1-1)d_2+i_2,(j_1-1)d_2+j_2} = x_{i_1 j_1} y_{i_2 j_2}$$

for all i_1, j_1, i_2, j_2 with $1 \le i_1, j_1 \le d_1$, and $1 \le i_2, j_2 \le d_2$. We find a possible matrix y by locating a non-zero entry $g_{i_0 j_0}$ in g, and setting y equal to the $d_2 \times d_2$ submatrix of g which contains the (i_0, j_0) position when g is naturally cut up into $d_2 \times d_2$ submatrices. More specifically,

$$y_{i_2 j_2} = g_{kd_2+i_2, ld_2+j_2}$$

for $1 \le i_2, j_2 \le d_2$, where $k = [(i_0 - 1)/d_2], l = [(j_0 - 1)/d_2]$. Then we define a $d_1 \times d_1$ matrix x by the rule

$$x_{i_1 j_1} = \frac{g_{(i_1-1)d_2+k_0,(j_1-1)d_2+l_0}}{g_{i_0 j_0}}$$

for $1 \le i_1, j_1 \le d_1$, where $k_0 = i_0 - kd_2, l_0 = j_0 - ld_2$. Either the equation

$$g_{(i_1-1)d_2+i_2,(j_1-1)d_2+j_2} = x_{i_1 j_1} y_{i_2 j_2}$$

holds, for all i_1, j_1, i_2, j_2 with $1 \le i_1, j_1 \le d_1$, and $1 \le i_2, j_2 \le d_2$, or there is no tensor decomposition of g with respect to the basis B.

4.3 The test ExtraSpecial

In this test it is required that S act absolutely irreducibly on the underlying vector space.

The test ExtraSpecial starts by factorising d. If d is not a prime power, r^m, or if the r-th power of some element of S is not scalar, the test returns false immediately.

Next we try to construct a sequence $x_1, y_1, x_2, \ldots, x_m, y_m$, of non-scalar elements of $\langle S \rangle$, which satisfy the following: x_i and x_j commute for all i, j, y_i and y_j commute for all i, j, x_i and y_j commute for distinct i, j, but the commutator of x_i and y_i is equal to z for all i, where z is a scalar element of $\langle S \rangle$, of order r. If at any stage the construction fails the test returns false.

First, we choose x_1 to be any non-scalar matrix in S, then choose y_1 to be any non-scalar matrix in S which does not commute with x_1, and define z to be the commutator of x_1 and y_1. If the order of z is not equal to r, then the test returns false. Otherwise x_1 and x_2 are marked as *selected*.

The remaining elements $x_2, y_2, \ldots, x_m, y_m$ are chosen one at a time as follows. Once $x_1, y_1, \ldots, x_i, y_i$ have been chosen for some $i < m$, a non-scalar element s of S is considered, which has not been already marked as selected. If one of the commutators $[x_j, s]$ or $[y_j, s]$, for $j \le i$, is not in $\langle z \rangle$, then we return false. Otherwise, we define integers u_j and v_j, less than r, for $j \le i$, by the equations $[x_i, s] = z^{u_i}$ and $[y_i, s] = z^{v_i}$. Then the element $sx_1^{v_1} y_1^{r-u_1} x_2^{v_2} \ldots x_i^{v_i} y_i^{r-u_i}$ must commute with all x_j and y_j with $j \le i$. Provided it is also non-scalar, we define it to be x_{i+1}. If this element is scalar, then we consider another element s, and repeat the procedure, and return false if we exhaust all possibilities for s before completing the construction of x_{i+1}. If we succeed in defining x_{i+1}, then we mark the element s that was used as selected.

The element y_{i+1} is chosen in much the same way. Another non-scalar element s of S, which has not been marked as selected, is considered, and an element s_0 is defined to be $sx_1^{v_1} y_1^{r-u_1} x_2^{v_2} \ldots x_i^{v_i} y_i^{r-u_i}$, provided this is non-scalar, and does not commute with x_{i+1}, where the integers u_j and v_j are chosen just as before. As before, if one of the integers u_j or v_j cannot be defined we return false. If $sx_1^{v_1} y_1^{r-u_1} x_2^{v_2} \ldots x_i^{v_i} y_i^{r-u_i}$ turns out to be scalar, or to commute with x_{i+1}, a different s is considered. If no suitable s_0

can be constructed in this way, the test returns false. Assuming that a non-scalar s_0 has been constructed, then, where $[x_{i+1}, s_0] = z^u$, y_{i+1} is set equal to $s_0^{u'}$, where u' is the multiplicative inverse of u mod r. If the commutator $[x_{i+1}, s_0]$ is not a power of z, then the test returns false. If we succeed in defining y_{i+1}, then we mark the element s that was used as selected.

It remains to verify that the elements $x_1, y_1, \ldots, x_m, y_m$ generate $\langle S \rangle$ mod Z, and that $\langle S \rangle Z$ is normal in G. We verify that the elements generate $\langle S \rangle$ by selecting each remaining non-scalar element s of S in turn, postmultiplying it by appropriate powers of each x_i and y_i as above until the resultant element s' commutes with each x_i and y_i, and then checking that the element is scalar. Similarly, we verify that $\langle S \rangle$ is normal by selecting each conjugate x of each element of $\{x_1, x_2, \ldots x_m, y_m\}$ by a generator of G in turn, postmultiplying x by appropriate powers of each x_i and y_i until the resultant element x' commutes with each x_i and y_i, and then checking that x' is scalar. If at any stage this procedure fails, the test returns false, otherwise G has been proved to normalise a group of order either r^{2m+1} or 2^{2m+2}.

4.4 The test SymTensorProduct

The test SymTensorProduct starts by factorising d. If d is not a proper power, the test returns false. Otherwise, for all pairs $r, m > 1$ such that $d = r^m$ we attempt to express V as a symmetric tensor product of m spaces of dimension r. If we find a decomposition of V corresponding to an embedding of G/Z in $PGL(r, q) \wr S_m$, then $\langle S \rangle^G$ should preserve each factor of the tensor product decomposition, and G should permute the factors.

We start by trying to express $V_{\langle S \rangle}$ as a tensor power, that is, we try to decompose V as a tensor product of m spaces of dimension r, each of which is preserved by $\langle S \rangle$. The procedure is naturally iterative. We try first to express V as a tensor product of two spaces of dimension r^{m_1} and r^{m_2}, and then to express each of those (if they have dimension greater than r) as a tensor product. At each stage we apply Smash on the appropriate $F\langle S \rangle$ module to help us to find such a decomposition (omitting Steps 10 and 11 of the algorithm). Of course Smash needs generators for a normal subgroup of $\langle S \rangle$ as input, so we have to supply those. This we do using the following random technique. Suppose that V has already been expressed as a tensor product $V_1 \otimes V_2 \otimes \cdots \otimes V_k$, but, say, V_k has dimension r^u, for some $u > 1$, and we want to try to write V_k as a tensor product. We simply select a sequence of N_{max} random elements s_i of S (for some predetermined limit N_{max}). Then we run Smash on the $F\langle S \rangle$-module V, with the set $\{s_i^{k_i}\}$ as input, where k_i is chosen so that some prime power of $s_i^{k_i}$ is scalar. (This is because, modulo scalars, G is contained in the direct product of its induced actions on the spaces V_i, and by choosing elements of a direct product in the manner just described, we are likely to find elements that act trivially on some factors, but not on others.) If for some s_i we find a decomposition of V_k as a tensor product $U_k \otimes W_k$ then we can write V as $V_1 \otimes \cdots \otimes V_{k-1} \otimes U_k \otimes W_k$, and so we continue to attempt to break it down completely as a tensor product of spaces of dimension r. If we fail to decompose at any stage after trying N_{max} random elements, then we give up. Thus it is clear that this test can only give a result in the case where the decomposition is found, and not otherwise. To resolve this, we would need a procedure which could determine conclusively whether or not a given module could be expressed as a tensor product, and currently we do not have any efficient such procedure.

Once the $F\langle S\rangle$-module V has been expressed as a tensor product $V_1 \otimes V_2 \otimes \cdots \otimes V_m$ of r-dimensional spaces over $GF(q)$, we need to try and construct the action of G on such a module.

Since the tensor product decomposition has been constructed via a series of binary decompositions, it is first necessary to reorder the basis to give a natural tensor basis $\{b_{i_1 i_2 \ldots i_m}\}$, from which the factors $V_1, V_2, \ldots, V_m$ can easily be identified.

Now for each generator g of G we attempt to find the action of g as a permutation on the factors of the tensor product $V_1 \otimes V_2 \otimes \cdots \otimes V_m$ by expressing it as a product of transpositions. Failure to do this shows that g has no such action on $\{V_1, V_2, \ldots, V_m\}$, and so in this case the test returns false. We have not however excluded the possibility of G permuting the factors of some other tensor product preserved by N, which we have failed to find.

More precisely, we define the $d \times d$ matrix π_{jk} to be the matrix which permutes the j-th and k-th factors of the tensor product by swapping pairs of basis elements $b_{i_1 \ldots i_j \ldots i_k \ldots i_m}$ and $b_{i_1 \ldots i_k \ldots i_j \ldots i_m}$. We postmultiply g by each of the matrices π_{1k_1} in turn until, for some k_1, the matrix of $g\pi_{1k_1}$ can be expressed as a Kronecker product of matrices x and y acting on the spaces V_1 and $V_2 \otimes V_3 \otimes \cdots \otimes V_m$, and so $g\pi_{1k_1}$ preserves the factor V_1. Considering now the action (represented by y) of $g\pi_{1k_1}$ on V_1 and $V_2 \otimes V_3 \otimes \cdots \otimes V_m$ we try to find k_2 such that $g\pi_{1k_1}\pi_{2k_2}$ preserves both factors V_1 and V_2, and then iterate to find k_3, k_4 etc. If we succeed in each stage of the iteration through the factors we have found the permutation action π_g of g on the factors V_i as a product of transpositions $(m, k_m) \ldots (2, k_2)(1, k_1)$ in S_m. If we fail it is because g does not permute the factors of this tensor product.

Once we have found such a permutation for every generator g we have proved that G preserves a symmetric tensor decomposition, and thus is a subgroup of an amalgamated wreath product of $GL(r, q)$ with S_m.

5 Termination and complexity

In this section, we consider the question of whether Smash will terminate, and we estimate the complexity of the algorithm. Since some steps in the procedure involve choosing random elements from the group until we find one with certain properties, it is theoretically possible that it will never terminate, because we could be unlucky with every choice. However, provided we can show that the proportion of elements with the required property is at least c, for some fixed $c > 0$, then the probability of choosing n elements without success is at most $(1 - c)^n$, which approaches 0 as n approaches infinity. We are assuming here that we are able to choose truly random elements of the group, and all of the arguments in this section are made under that assumption.

We shall show that the probabilistic complexity of Smash is bounded by a polynomial function of the dimension d of the group. This means that there is a positive integer n such that, for any given $\varepsilon > 0$, there is a constant K such that Smash will terminate within time Kd^n with probability at least $1 - \varepsilon$. The constant K will depend on the number of generators of G and the initial number of generators of S, which we are assuming to be fairly small constants. It will also depend on the order q of the field, but only to the extent that the basic field operations are, and the complexity here is $\log(q)$, which is not significant for reasonably small fields. (In fact, in GAP, field operations are carried out using lookup tables.)

Our worst case analysis produces the value of 6 for n, which is much worse than we would like. When designing matrix group algorithms, we always aim for the complexity $O(d^3)$, which is the same as that of matrix multiplication. One of the factors d in our $O(d^6)$ estimate arises from the fact that we could conceivably have to apply the MEATAXE as many as d times in Step 1 to find an irreducible $F\langle S\rangle$-submodule of V. (The steps of the algorithm are as listed in Section 3.) In fact this is highly improbable (although we have not attempted to estimate the precise probability). Experience shows that at most two or three MEATAXE calls are necessary in practice. (The fact that V is completely reducible as an $F\langle S\rangle^G$-module (that is, decomposes as a direct sum of irreducible submodules), tends to decrease the number of calls required.) Another factor of d^2 in the complexity arises from the fact that a large number of iterations of the main SMASH loop could theoretically be necessary. So, if we could find some heuristic means of keeping this number down to constant size, then, for practical purposes, we would have got the complexity down to $O(d^3)$, which is what we are aiming for. There is some discussion on the best ways of achieving this in Section 6.

Turning now to the precise analysis, let us first consider how many times we may have to go through the main loop. With each iteration, we adjoin one more element to S, and this element either does not fix the subspace W, in which case the length of an $F\langle S\rangle$-composition series of V decreases, or it fails to centralise the matrix C (in Steps 7 and 9), in which case the centralising field of one of the composition factors decreases. Clearly, the first possibility can happen at most $d-1$ times altogether. When the second possibility occurs, the order of the centralising field of some $F\langle S\rangle$-composition factor of dimension f is reduced from q^e to $q^{e'}$, for some integers e, e', f, where e divides f and e' divides e. Since the sum of the dimensions of all composition factors is d, it is not hard to see that the number of times that this could happen altogether is also bounded above by d. This argument puts an upper bound of $2d$ on the number of iterations. It has the unfortunate consequence that we have to assume that the set S has size $O(d)$. In fact, there do exist examples in which $d-1$ iterations occur in practice. We find one by choosing $G = GL(d,q)$ and letting S initially contain just one element A, which is a diagonal matrix with all entries 1 or 0 except for A_{11}.

For each iteration of the main loop, we have to go through the steps listed in Section 3. The worst of these is Step 1. As we have already remarked, we may require up to $d-1$ iterations of the MEATAXE. We use the algorithm and implementation that are described in [7]. There is a configuration described there for which the algorithm fails to terminate. Since this is highly improbable, we shall ignore it. We are justified in doing this, because the configuration cannot arise when the module is completely reducible and we know that V is indeed completely reducible as an $F\langle S\rangle^G$-module. We could therefore escape from the configuration by adding random conjugates to S. With this exception, the MEATAXE has probabilistic complexity $O(d^3)$ for a fixed set of generating matrices. However, since we have to assume that S has size $O(d)$, this complexity increases to $O(d^4)$, so we end up with complexity $O(d^5)$ for Step 1.

The remaining steps up to Step 9, which are the ones that form the components of the main SMASH loop, can all be seen to have complexity at worst $O(d^4\log(d))$, so we shall just go through them briefly. Since S has size $O(d)$, it is straightforward to see that Step 2 has complexity $O(d^4)$. Step 3 does not depend on S, but uses the generators of G and a given subspace. The main loop of the MinBlocks algorithm is to calculate the images of a basis of V under the generators of G, and express the result in terms of the basis using Gaussian elimination. This has complexity $O(d^3)$. Other parts of the procedure involve amalgamating and renumbering blocks. Since there can be at most $d-1$ block amalgamations altogether, this has complexity at worst $O(d^2)$. Step 4 is another $O(d^4)$ process, since the size of S is $O(d)$. In Steps 5 and 8, we may conceivably have to consider all divisors of d in our search for the centralising field, so this could have complexity $O(d^4 \log(d))$. Step 6 just involves scanning the entries of the generators of G, and so has complexity $O(d^2)$. In Steps 7 and 9, the matrix C has to be raised to the power q^e, where $e \leq d$. This could involve up to $d \log(q)$ matrix multiplications, and so these steps have complexity $O(d^4 \log(q))$.

We stress that the theoretical complexity of these components does not necessarily give an accurate indication of which are the most expensive steps in practice, for groups of small degree. As we shall see in Section 6, for examples of degree up to about 100, Step 1 really does seem to take up the bulk of the time, whereas Steps 5, 7, 8 and 9, which are theoretically slow, are fast in practice.

Most of the steps can conclude with a search for a random conjugate of an element of S that either does not fix W, or does not centralise the matrix C. Since the elements of $N = \langle S \rangle^G$ that do, respectively, fix W or centralise C form a subgroup of N, at least half of the elements of N must have the sought-after property, so we can expect to find a suitable conjugate quickly. Calculating the conjugate and testing the property each have complexity $O(d^3)$.

The final two steps, 10 and 11, can only happen once, and so they are not affected by the number of times we go through the main loop. For the extraspecial case (Step 10), we have $d = r^m$, where $N = \langle S \rangle^G$ has order r^{2m+1} or 2^{2m+2}. Any irredundant generating set for N has size $2m+1$ or $2m+2$, respectively, and since whenever we add a new element to S, we always make $\langle S \rangle$ bigger, it follows that our final set S of generators for N will have at most $2m+1+t$ elements, where t is the initial size of S. For each pair of generators of N, we have to form their commutator, identify the resulting power of the central element z, and multiply by a power of a generator. The last of these has the highest complexity, which is $O(d^3 \log(r))$. Similar considerations apply to the calculation of the conjugation action of G on N, where we form the commutators of the generators with their conjugated images under each generator of G. Thus the whole process has complexity at worst $O(m^2 d^3 \log(r))$, which is asymptotically less than $O(d^4)$.

Step 11 is theoretically the slowest of all, since it involves recursive calls of Smash. However, in the recursive calls we omit Steps 10 and 11 of the algorithm (since we are only interested in inhomogeneous tensor product decompositions), so the recursion causes no problem with termination. If $d = r^m$ and we are looking for m tensor factors of degree r, then we will need $m-1$ successful recursive calls to find the full decomposition, and for each of these, we may have up to N_{max} unsuccessful recursive calls. However, all but the first of the tensor product decompositions sought will be on spaces of dimension $r^{m'}$ for $m' < m$, and so in fact the total complexity of the recursion is still only a constant times the complexity of Steps 1 - 9 of Smash. Thus Step 11 does not increase the total complexity of Smash, it merely multiplies it by a constant. Once the decomposition has been found, it has to be verified by computing the permutation action of the generators of G on the tensor factors. This can be seen to have complexity about $O(m^2d^3)$, which is small in comparison with the recursive part of Step 11.

6 Performance and possible improvements

The six tables below give CPU times for various runs of SMASH. The different tables correspond to the possible outcomes of these runs. In the first, SMASH finds no decomposition, and in the remaining five, it finds one of the five possible decompositions. Each CPU time is in seconds, and is followed by two numbers in brackets. The first of these is the initial size of the set S given to SMASH as input, and the second is the number of times that SMASH needed to go through the main loop; for each cycle through this loop after the first, a new element is added to S, and so the final size of S is one less than the sum of these two numbers.

These tests were carried out on a Silicon Graphics Iris WorkStation. The precise times should in any case not be taken too seriously, since they can vary considerably from run to run, due to random aspects of the algorithm. Furthermore, our implementation in GAP is a prototype, which makes no claims to particular efficiency. The times are of interest mainly from the point of view of comparing them with each other.

Some of the names of the groups in these examples are reasonably self-explanatory (such as $3J_3$, which is the 3-fold central cover of the sporadic simple group J_3) and follow the notation of the *Atlas* [4], whereas others need defining. The group E_1 is the direct product of an extraspecial group of order 3^5 with the group $\Gamma L(1,7^7)$. The module is a tensor product of modules of dimension 9 and 7 for these groups over the field $GF(7)$. This group is a subgroup of $\Gamma L(9,7^7)$ and so it is semilinear. It is also imprimitive (since the first direct factor is) and of course it is a tensor product, so it appears in three of the tables. The decomposition found by SMASH depends on the set S, but it can also differ from run to run with the same input, due to random aspects in the algorithm. The group E_2 is the direct product of the quaternion group Q_8 with $\Gamma L(25,3^2)$, and the module is a tensor product of modules of dimensions 2 and 50 for these groups over the field $GF(3)$. This group is also semilinear and imprimitive. The group E_3 appearing in Table 4 is a direct product of $L_3(2)$ and $L_3(3)$ acting on a tensor product of modules of dimensions 3 and 26 for these groups over $GF(2)$. The groups N_{r^n} in Table 5 are normalisers of extraspecial groups or groups of symplectic type of order r^n.

The number d in the tables is the dimension of the representation. The number r in Table 3 is the number of blocks in the system of imprimitivity found by SMASH, and the number r in Table 4 is the dimension of the isomorphic irreducible modules found for the normal closure of S. So r and d/r are the degrees of the modules in the tensor product decomposition found by SMASH.

The results in Table 1, in particular, seem to indicate that it is preferable not to start with too small a set S. The timings in the first line have S initially containing a single involution, and those in the second line start with a single random element. In the third line, we start with four conjugate involutions, and in the fourth line four random elements in the normal closure of the initial involution. It seems preferable not just to enlarge the set S by random conjugates of its initial elements, but to use random conjugates of products of its elements. Of course, we do not wish to make S

G	$6A_7$	M_{12}	$3J_3$	$L(3,5)$	M_{22}
d	24	55	80	124	154
Time	14.0 (1,5)	29.5 (1,4)	57.6 (1,3)	155.9 (1,3)	287.3 (1,4)
	6.1 (1,2)	11.3 (1,2)	39.1 (1,2)	62.0 (1,2)	145.0 (1,2)
	8.7 (4,2)	23.0 (4,2)	30.2 (4,1)	53.7 (4,1)	138.8 (4,1)
	5.3 (4,1)	9.1 (4,1)	23.0 (4,1)	39.2 (4,1)	156.3 (4,1)

Table 1: SMASH returns *false*

G	$\Gamma L(10,3^2)$	$\Gamma L(10,8^5)$	E_1	E_2	$\Gamma L(1,19^{100})$
d	20	50	63	100	100
Time	27.9 (2,10)	114.4 (2,10)	20.2 (5,1)	44.9 (3,1)	183.4 (1,1)
	8.5 (5,2)	42.3 (5,1)			

Table 2: SMASH returns *semilinear*

unnecessarily large, and the size four seems to be a reasonable compromise for these sorts of examples. Of course, in some examples, particularly those in Tables 5 and 6, the smallest cardinality of a generating set of the normal closure of S is quite large, and so it is inevitable that we will end up making several passes of the main SMASH loop.

Although the tables do not indicate this, in general a large proportion of the total time is taken up with testing modules for irreducibility, using the MEATAXE algorithm (well over half of the time in the examples in Table 1). In some of the larger examples, the time taken to express the module as a sum of translates of an irreducible $\langle S\rangle$-module can also be large. These processes can certainly be made to run much faster with more efficient implementations; in the C language, for example. The procedure MINBLOCKS used to find blocks of imprimitivity is relatively inexpensive. A possible variation of the algorithm would be to apply Step 3 to W before, rather than after, Step 2 (see Section 3). This will usually be less fast if the group is primitive, particularly if a large number of passes through the main SMASH loop are necessary, but it can lead to dramatic improvements in the imprimitive case. In the bad example that takes 1706 seconds in Table 3, if we apply this policy then the CPU time can decrease to as little as 50 seconds. The example taking 539 seconds in Table 4 increases to 671 seconds with this policy, but if we increase the initial size of S to 4 (according to the policy advocated above), then the time goes down to 166 seconds.

A tentative conclusion is that if the set S has initial size less than 4, then its size should be increased to about 4 by adding random elements of the normal closure of S before starting the main algorithm, and that the option should be available to apply Step 3 to W before Step 2. It is difficult to make definitive decisions about these matters, since performance can vary so much from example to example.

G	$A_8 \wr P_6$	$M_{11} \wr M_{11}$	E_1	E_2	$6A_7 \wr C_7$	$6A_7 \wr C_7$
r	6	11	3	2	7	168
d	24	55	63	100	168	168
Time	24.1 (1,4)	30.9 (1,2)	14.4 (2,1)	33.7 (1,3)	1706 (1,8)	161.7 (1,1)

Table 3: Smash returns *imprimitive*

G	E_1	E_2	E_3	E_3	E_2	$M_{22} \times N_{3^5}$
r	9	7	3	26	50	21
d	63	63	78	78	100	189
Time	16.7 (2,1)	20.0 (4,1)	52.5 (1,2)	23.3 (2,1)	63.3 (2,1)	539.1 (1,4)

Table 4: Smash returns *tensor product*

G	N_{2^6}	N_{7^3}	N_{2^8}	N_{3^5}	$N_{2^{12}}$	N_{3^9}
d	4	7	8	9	32	81
Time	5.7 (1,4)	0.7 (1,2)	2.6 (1,6)	2.3 (1,4)	30.2 (1,10)	374.5 (1,8)

Table 5: Smash returns *extraspecial*

G	$J_1 \wr C_2$	$N_{2^6} \wr S_3$	$GL(3,17^2) \wr S_4$	$SL(2,2^{12}) \wr L(3,2)$
d	$7^2 = 49$	$4^3 = 64$	$3^4 = 81$	$2^7 = 128$
Time	117.2 (1,6)	503.8 (2,9)	584.0 (1,14)	1539.2 (1,14)

Table 6: Smash returns *symmetric tensor product*

References

[1] M. Aschbacher, On the maximal subgroups of the finite classical groups, *Invent. Math.* 76 (1984) 469-514.

[2] John Cannon and Wieb Bosma, *Handbook of* Magma *functions*. Department of Pure Mathematics, Sydney University, 1993.

[3] A.H. Clifford, Representations induced in an invariant subgroup, *Ann. of Math.* 38 (1937), 533-550.

[4] J.H. Conway, R.T. Curtis, S.P. Norton, R.A. Parker and R.A. Wilson, Atlas of finite groups, Clarendon Press, Oxford 1985.

[5] Larry Dornhoff, Group Representation Theory, part A, Marcel Dekker Inc. 1971, New York.

[6] Derek F. Holt, C.R. Leedham-Green, E.A. O'Brien and Sarah Rees, Testing matrix groups for primitivity, *preprint.*

[7] Derek F. Holt and Sarah Rees, Testing modules for irreducibility, *J. Austral. Math. Soc. Ser. A*, 57, 1994.

[8] C.R. Leedham-Green, A non-constructive recognition algorithm for SL, *preprint.*

[9] C.R. Leedham-Green, Recognising the Special Linear Group, *preprint.*

[10] Peter M. Neumann and Cheryl E. Praeger, A recognition algorithm for special linear groups, *Proc. London Math. Soc.* 65 (1992), 555-603.

[11] R. Parker, 'The computer calculation of modular characters. (The Meat-Axe)', in *Computational Group Theory*, (ed. M.D. Atkinson) Academic Press, London, 267-74, 1984.

[12] Cheryl E. Praeger, Computation with matrix groups over finite fields, in *Groups and Computation, Workshop on Groups and Computation, October, 1991*, ed. Larry Finkelstein, William M. Kantor, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 11, American Mathematical Society, 189-196, 1993.

[13] Martin Schönert *et al.*, GAP – *Groups, Algorithms and Programming.* Lehrstuhl D für Mathematik, RWTH, Aachen, 1993.

Derek F. Holt
Mathematics Institute
University of Warwick
Coventry CV4 7AL
Great Britain

E-mail: dfh@maths.warwick.ac.uk

C.R. Leedham-Green
School of Mathematical Studies
Queen Mary and Westfield College
University of London
Mile End Road, London E1 4NS
Great Britain
C.R.Leedham-Green@qmw.ac.uk

E.A. O'Brien
School of Mathematical Sciences
Australian National University
Canberra, 0200
Australia
E-mail: Eamonn.OBrien@maths.anu.edu.au

Sarah Rees
Department of Mathematics and Statistics
University of Newcastle
Newcastle-upon-Tyne NE1 7RU
Great Britain
Sarah.Rees@newcastle.ac.uk

Last revised January 14, 1994