Testing matrix groups for primitivity

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

Abstract

We describe an algorithm which seeks to decide whether or not a matrix group defined over a finite field acts to preserve blocks of imprimitivity and, if so, to find a block system. An implementation of the algorithm is publicly available.

1 Introduction

In this paper we present an algorithm which seeks to decide whether or not a matrix group of finite dimension over a finite field preserves a non-trivial system of blocks of imprimitivity. In most cases, our algorithm will either find that the group is imprimitive, and return at least one block system, or it will prove that the group is primitive. Occasionally, it may discover that the group is semilinear over an extension field before resolving the primitivity question, in which case it does not conclusively settle whether or not the group is imprimitive.

Let $q = p^m$, where p is prime and $m \geq 1$; let V be the vector space F_q^d of row vectors on which the general linear group, $GL(d, q)$, acts.

Assume G is a subgroup of $GL(d, q)$ and that G acts irreducibly on V. Then G acts imprimitively on V if there is a non-trivial direct sum decomposition

$$V = V_1 \oplus V_2 \oplus \ldots \oplus V_r$$

where $V_1, \ldots, V_r$ are permuted by G. In such a case, each block V_i has the same dimension or size, which we shall denote by s, and we have the *block system* $\{V_1, \ldots, V_r\}$. If no such system exists, then G is primitive. We use r and s throughout the paper to denote the number and size of blocks, respectively.

The algorithm described in this paper is another contribution to the "recognition project" for matrix groups defined over finite fields. The theoretical framework for much of this project is provided by the Aschbacher [1] classification of subgroups of $GL(d, q)$, where one of the nine "categories" is that the group acts imprimitively. Algorithms have already been developed to recognise other categories. These include the MEATAXE algorithms of Parker [9] and Holt & Rees [6] to recognise reducible groups; and the Neumann & Praeger [8] recognition algorithm for groups containing the special linear group.

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

We set as our goal to develop a "practical" algorithm which given as input a matrix group, described by a generating set, of dimension up to about 100 over "moderate" fields, can decide whether or not the group is primitive.

In conjunction with its development, the algorithm was implemented in the computational group theory system GAP [11]. Our desire for a "practical" algorithm significantly influenced its final structure. an implementation will also be available within MAGMA [2].

Recall that G is *irreducible* if there is no non-trivial proper subspace of V invariant under G, and that G is *absolutely irreducible* if it remains irreducible under any extension of the ground field. When G is irreducible but not absolutely irreducible, there is an extension field $E = GF(q^e)$ of F, where e divides d, and V can be regarded as a vector space of dimension d/e over E, with G acting linearly over E. More generally, we are interested in the case in which, for some such extension field E, the group G acts semilinearly on V regarded as an E-space, where the field automorphisms which occur fix F. That is to say, there is a homomorphism α of G into the Galois group of E over F such that $(\lambda v)^g = \lambda^{g\alpha} v^g$, for all $v \in V$, all $g \in G$, and all $\lambda \in E$. In this case, we shall say that G is semilinear of degree e. If G is semilinear of degree e for some $e > 1$, then we shall simply say that G is semilinear.

The *projective order* of an element g of G is the least positive integer o such that g^o is a scalar matrix.

Our primitivity algorithm takes as input a generating set for a matrix group G, which is assumed to act irreducibly on V. Three possible conclusions can be reached by the algorithm:

(i) it decides that G is semilinear;

(ii) it decides that G is imprimitive, and returns one block system;

(iii) it decides that G is neither semilinear nor imprimitive.

It is important to note that conclusion (i) does not prove that G is primitive, and conclusion (ii) does not prove that G is not semilinear. However, each conclusion effectively reduces the "recognition problem" to a simpler problem.

In summary, our primitivity algorithm proceeds as follows. If G is not absolutely irreducible then, as we saw above, it is semilinear. Hence, we first decide whether or not G is absolutely irreducible; if not, we terminate. Otherwise, we carry out a test that will either prove that G is semilinear, or that G has an explicit block system on which it acts as an abelian group, or that neither condition is satisfied; it is only in the last case that the algorithm continues. The next step is to consider various cyclic subgroups of G. These may resolve the issue by consideration of their order, which can only produce a negative answer, or by consideration of their action, which may produce a negative answer, or lead us to a block system. Finally, we consider the actions of non-cyclic subgroups of G in a series of tests which may again rule out the existence of a block system or find one.

Our methods for proving the primitivity of G, or for finding a block system, do not constitute an algorithm that will provably answer the question in all cases. Even if we can decide that G is not semilinear, then it is theoretically possible that the algorithm will never terminate, or that it will not do so in an acceptable period of time. However, we know of no example where the performance of the algorithm is unacceptable for degrees up to 100 over "moderate" fields; see the performance tables at the end for details, and note that some of these cases have degrees well over 100. The fine tuning of the algorithm will depend on the speed of its various components, and our first implementations of these components can no doubt be speeded up by varying amounts.

A key component of our test for primitivity is the algorithm encoded as the procedure MinBlocks – given a non-trivial subspace of a block, this algorithm finds the block system with minimal block size s that contains this subspace.

Another key component is the algorithm encoded as the procedure Smash. It is described in [4]. In summary, given a set S of elements of a matrix group G, this algorithm investigates whether G has certain decompositions with respect to the normal closure, $\langle S\rangle^G$, of $\langle S\rangle$ in G.

The primitivity algorithm consists of a sequence of tests of increasing cost and complexity. Usually, a test is premised on the assumption that a block system exists for some particular values of r and s. In practice, the test then seeks either to rule out the existence of a block system of this particular size, or to find a subspace of a block to supply to MinBlocks. Such a call to MinBlocks may, in fact, find a block system having a size different from s.

This paper is organised as follows. We first describe the important components of our algorithm. In Sections 3, 4 and 5, we present the steps of the algorithm. Finally, we report on its implementation, and comment on its effectiveness and performance.

2 Components of our algorithm

In this section, we describe each of the components, MinBlocks and Smash, in turn. We also discuss an important feature of the Meataxe algorithm.

In Section 6, we mention other essential components of our primitivity algorithm: selecting random elements, and computing their orders, projective orders, and characteristic polynomials.

2.1 The MinBlocks algorithm

The algorithm encoded as MinBlocks has some parallels with the coincidence procedure of a coset enumerator.

Let $\mathcal{S}$ be a collection of subspaces of V. We call an element W of $\mathcal{S}$ *independent* if there is no subset of $\mathcal{S}$, not containing W, whose sum intersects W non-trivially.

The algorithm takes as input a set $\mathcal{S}$ consisting of a single subspace of V. At each stage in its application, the elements of $\mathcal{S}$ are independent subspaces of V. We compute the image of each subspace in $\mathcal{S}$ under the given generating set of G. If the subspaces in $\mathcal{S}$ are permuted by the generators, then they form a block system as required. If not, a subspace $W \in \mathcal{S}$ and an element g of G are found such that $Wg \notin \mathcal{S}$. If Wg is independent of the other spaces in $\mathcal{S}$, it is added to $\mathcal{S}$. Otherwise, a minimal subset $\mathcal{T}$ of $\mathcal{S}$ is found such that Wg is *dependent* on $\mathcal{T}$ – that is to say, such that Wg intersects the (direct) sum of the elements of $\mathcal{T}$ non-trivially. Then $\mathcal{T}$ is removed from $\mathcal{S}$ and replaced by $\sum_{U\in\mathcal{T}} U + Wg$. This process is continued until either $\mathcal{S}$ consists of the single space V, or a block system is found. In practice, however, the procedure can halt as soon as an element of $\mathcal{S}$ has dimension greater than half the dimension of V. It can be speeded up if, whenever a space is replaced by a space of larger dimension, all the other spaces in $\mathcal{S}$ are discarded.

2.2 The Smash algorithm

The algorithm encoded as the Smash procedure is discussed in detail in [4]. Here, for completeness, we summarise the details of the algorithm. It is of interest in its own right, but here we focus on its application to primitivity testing.

Assume that the matrix group G acts absolutely irreducibly on the space V. The input to Smash consists of generators for G and a set of matrices S, not all of which are scalar. Then Smash investigates whether G has one of the following types of decompositions with respect to $N = \langle S \rangle^G$:

1. G acts imprimitively on V, with blocks $V_1, V_2, \ldots, V_r$, and N fixes each block;
2. G preserves a tensor product decomposition $U \otimes W$ of V, and the induced action of N on U is scalar;
3. G is semilinear of degree e, for some divisor e of d with $e > 1$, and N acts linearly on V regarded as a vector space over $GF(q^e)$;
4. G preserves a symmetric tensor product decomposition of V, where N preserves the tensor factors;
5. G has a normal subgroup M with $MZ(G) = NZ(G)$, where M is either an extraspecial group of odd prime-power order or a 2-group of symplectic type.

The investigation is conclusive with one exception – in its current form, Smash may fail to discover that G preserves a symmetric tensor product.

In Smash, we eventually construct a decomposition of V as a direct sum, $V_1 \oplus V_2 \oplus \ldots \oplus V_t$, of irreducible FN-modules. Four situations arise:

- If $t > 1$ and the V_i are not pairwise isomorphic as FN-modules, then G acts imprimitively on V (decomposition type 1).
- If $t > 1$ and the V_i are both pairwise isomorphic and absolutely irreducible as FN-modules, then V can be recognised as a tensor product preserved by G (decomposition type 2).
- If the V_i are pairwise isomorphic and not absolutely irreducible as FN-modules, then G is semilinear (decomposition type 3).
- Otherwise, $t = 1$ and N acts absolutely irreducibly on V. (In its general application, Smash now seeks to determine whether G has a decomposition of type 4 or 5 – in our restricted context of primitivity testing, we terminate.)

The application of Smash to primitivity testing occurs as follows. If G has a block system containing r blocks of size s, then there is a homomorphism from G to S_r.

In each test, we choose a particular value of r and assume that there is such a block system. With this assumption, during the application of the test, we may discover that a particular non-scalar element of G must lie in the kernel of the homomorphism from G to S_r.

If we find such an element, g, we seek to build up its normal closure, $\langle g \rangle^G$, in G and then determine which of our four conditions applies.

In practice, a set S is initialised to contain g. Then S is supplied to SMASH, which seeks to satisfy one of the four conditions under the assumption that $\langle S \rangle = N$. If none is satisfied, then $\langle S \rangle$ cannot be normal in G, and SMASH seeks to build up the normal closure of $\langle S \rangle$ under G by adding random conjugates to S; it then applies relevant parts of the procedure to this larger set. Eventually, SMASH will terminate when one of the conditions is satisfied (which could conceivably occur before $\langle S \rangle = N$).

If SMASH discovers that G is semilinear, it is not currently possible to settle conclusively whether or not G acts imprimitively.

If SMASH discovers that G preserves a tensor product $U \otimes W$ of V with the induced action of N on U scalar, then we seek to decide whether or not G acts imprimitively on the first component, U, of the tensor decomposition. If the action of G on U is imprimitive, then G is also imprimitive in its action on V, and a block system for V can be constructed from that found for U. Conversely, suppose that, in this situation, G has a block system in which $g \in G$ is non-scalar and g fixes all blocks. Then $N = \langle g \rangle^G$ must also fix all blocks. But, as we saw above, N preserves a decomposition of V as a direct sum of irreducible FN-submodules isomorphic to W. The blocks must therefore be sums of s' such subspaces for some s', and it follows that G is imprimitive on U with blocks having size s'.

If SMASH discovers that N acts absolutely irreducibly on V, then g cannot fix a block system containing r blocks, and so we can rule out r.

In summary, a call to SMASH ensures that we learn that G is semilinear and we can draw no conclusion about its primitivity; or we find a block system; or we rule out some of the possible block sizes.

2.3 The MEATAXE algorithm

Let F be a finite field and G a finite group. The MEATAXE is an algorithm for deciding whether or not an FG-module is irreducible and, where it is reducible, for finding an explicit submodule. It was first implemented and described by Parker [9], using ideas of S.P. Norton. Since then, it has become a standard tool in computational group theory, and there have been several efficient implementations. The original version was designed for small fields (and mainly for groups that are close to being simple), and its efficiency decreases sharply as the size of the field increases. A version which does not suffer from these deficiencies is described in Holt & Rees [6], and we use their GAP implementation of this algorithm in primitivity testing. Both the original and the Holt–Rees version can decide whether or not two irreducible FG-modules are isomorphic.

Thus, for a general FG-module V, it is possible to identify its composition factors, and determine their multiplicities in V. Since the calls to the MEATAXE are recursive, the composition factors are not in general found as submodules of V (but rather of some quotient of V). However, where U is a composition factor of V, it is also possible to use the MEATAXE machinery to find a submodule W of V, containing U as a composition factor of nonzero multiplicity m, with the property that no proper FG-submodule of W has U occurring as a composition factor with the same multiplicity m. We shall denote such a submodule by V_U. In general, V_U need not be unique, but we shall see that it is unique in the case when U has multiplicity one in V.

Since we make use of this facility, and it does not appear to have been described elsewhere, we describe it here. It is not a new idea however. A similar technique is used in the Lux and Ringe implementation of the MEATAXE, which can find all FG-submodules of V. This implementation, developed at Aachen, is described in Ringe [10].

Let $G = \langle g_1, \ldots, g_n \rangle$, and let V be an FG-module defined by matrices $A_1, \ldots, A_n$ corresponding to the g_i. Let R be the ring of polynomials over F in the non-commuting variables $x_1, \ldots, x_n$. Then, for an FG-module V, there is a ring homomorphism ϕ_V from R to the algebra generated by the matrices A_i, defined by $x_i \to A_i$ for $1 \leq i \leq n$. In the MEATAXE, we attempt to find $\theta \in R$ such that $\phi_V(\theta)$ has small but non-trivial nullspace $N_V(\theta)$. For certain selected $v \in N_V(\theta)$, we then use the so-called *spinning* process to calculate the minimal FG-submodule $\langle v \rangle FG$ of V that contains v. Using this technique on V and on its dual, we either find an explicit submodule, or we obtain enough information to deduce theoretically that V is irreducible; see Holt & Rees [6] for further details.

Assume that we have used these techniques to find the distinct composition factors $U_1, \ldots, U_t$ of V with multiplicities. We then find elements θ_i of R such that $N_{U_i}(\theta_i)$ is nonzero, but $N_{U_j}(\theta_i)$ is zero for $i \neq j$. We do this by considering random elements of R. It can be shown by probabilistic arguments that we can expect to find suitable θ_i reasonably quickly. Since U_i is a composition factor of V, $N_V(\theta_i)$ must be nonzero for each i. Choose $v_i \in N_V(\theta_i)$ with $v_i \neq 0$. We claim that $V_{U_i} := \langle v_i \rangle FG$ has the required minimality property. Since $v_i \in N_V(\theta_i)$ and $v_i \in V_{U_i}$, we have $v_i \in N_{V_{U_i}}(\theta_i)$, which is therefore nonzero. As $N_{U_j}(\theta_i) = 0$ for $j \neq i$, this implies that the multiplicity m of U_i in V_{U_i} is nonzero. Suppose that X is a submodule of V_{U_i} which has U_i with the same multiplicity m. Then, again using the fact that $N_{U_j}(\theta_i) = 0$ for $j \neq i$, we see that $N_{V_{U_i}}(\theta_i) = N_X(\theta_i)$. But then $v_i \in N_X(\theta_i)$ and so $v_i \in X$, which forces X to equal V_{U_i}. This establishes the claim.

If U has multiplicity 1 in V, then the intersection of any two submodules V_1 and V_2 of V that have U as a composition factor must itself have U as a composition factor, for otherwise U would have multiplicity two in $(V_1 + V_2)/(V_1 \cap V_2)$. It follows that V_U is the unique minimal such submodule in this situation.

3 Reduction tests

As the first step of the primitivity algorithm, we apply two reduction tests to the supplied group.

The input to the algorithm is a generating set of matrices, $g_1, \ldots, g_n$ for a group G. Let V be the FG-module defined by these matrices. In our subsequent discussion, we assume that V has dimension at least 2.

We first decide whether or not G is absolutely irreducible by a call to the MEATAXE. If G is not absolutely irreducible, then it can be written as a module of smaller dimension over a larger field – in fact, G is semilinear in this case, and we terminate the primitivity algorithm.

We next seek to decide in general whether or not G is semilinear, by a call to SMASH. Let G' denote the derived group of G. If G is semilinear, then V has a direct sum decomposition as isomorphic irreducible FG'-modules V_i, and G' does not act absolutely irreducibly on the V_i.

We first construct a normal generating set for G' by initialising the set S to contain all of the commutators of the generators of G.

If S does not consist entirely of scalars, we now call SMASH with input S. If G is semilinear, then $\langle S\rangle^G$ will not act absolutely irreducibly on the V_i, and so SMASH will either find a block system or conclude that G is semilinear.

If S consists entirely of scalars, we add a non-scalar generator of G to S. Note that G must have a non-scalar generator – otherwise, it is reducible and will be eliminated by the MEATAXE call. The addition of one non-scalar generator of G to S ensures that $\langle S\rangle$ is a normal abelian subgroup of G, and so it cannot act absolutely irreducibly on V. Further, V cannot decompose into a direct sum of isomorphic absolutely irreducible $F\langle S\rangle$-modules V_i: since $\langle S\rangle$ is abelian, an irreducible $F\langle S\rangle$-module must be 1-dimensional, and, by Schur's lemma, if two such are absolutely irreducible and isomorphic, $\langle S\rangle$ must be scalar. We call SMASH with input S, and, as before, SMASH will either find a block system or conclude that G is semilinear.

If we find a block system or deduce that G is semilinear, we terminate the algorithm. Otherwise, we conclude that G is not semilinear and proceed to the next test.

4 Investigating actions of cyclic subgroups

In this section, we describe those tests which use various cyclic subgroups of our supplied group to find a block system, or rule out the existence of block systems having particular block sizes.

4.1 Element orders

If G has a block system consisting of r blocks of size s, then G has an embedding in $GL(s,q) \wr S_r$, where S_r is the symmetric group on r points.

This observation facilitates the following test. Assume that g is a element of G, having order o. Does o divide the exponent of $GL(s,q) \wr S_r$? If not, then we can rule out r and s.

In practice, we first compute the exponent of $GL(s,q)$ using the following observation. Let k be the smallest integer which satisfies the inequality $p^k \geq s$; then the exponent, e, is $p^k \times \mathrm{lcm}(q-1, q^2-1, \ldots, q^s-1)$.

We next select a random element, g, of G and compute its order, o. If G embeds in $GL(s,q) \wr S_r$, then there is an element of S_r having order $o/\gcd(o,e)$. We now use the following simple test: if S_r has an element of order $p_1^{n_1} p_2^{n_2} \ldots p_k^{n_k}$, where the p_i are distinct primes and $n_i > 0$, then $p_1^{n_1} + p_2^{n_2} + \ldots + p_k^{n_k} \leq r$.

The order test is inexpensive to apply and is also extremely effective in eliminating values of s.

4.2 Characteristic polynomial structure

Assume that g is an element of prime order p. As before, we assume that there exist r blocks of size s, for some chosen values of r and s. Then g acts to permute the r blocks, which are organised into cycles of length p or remain fixed.

Consider a single p-cycle of blocks of size s under the action of g. Then, with respect to an appropriate basis, g acts on this p-cycle as s copies of a $p \times p$ permutation matrix of order p. The characteristic polynomial, f, of g must have the form

$$f(x) = (x^p - 1)^{su} \times R(x) \tag{1}$$

where u is the number of p-cycles and $R(x)$ is the characteristic polynomial of the restriction of g to its action on the blocks that it fixes.

If G has r blocks of size s, there is an embedding of G in $GL(s,q) \wr S_r$. Hence, if the order of g does not divide the order of $GL(s,q)$, then g can act on the fixed blocks only as the identity. In these cases, the characteristic polynomial of g must have the simpler structure

$$f(x) = (x^p - 1)^{su} \times (x-1)^{sc} \tag{2}$$

where c is the number of fixed blocks.

We first consider the case where p differs from the characteristic of F. Given a non-scalar matrix g of order p, we compute its characteristic polynomial, f, and use it to formulate the following test.

1. Find the largest power, t, of $x^p - 1$ which divides f. We call t the *free-rank* of g.
2. Establish whether the remainder, $R(x)$, is simply a power of $(x-1)$. If not, set a variable, Excess, to true.
3. If p does not divide the order of $GL(s,q)$, then $f(x)$ must have the structure outlined in Equation (2). But, if Excess is true, then the polynomial does not have the required factorisation, and we can rule out s as a block size.
4. If p does divide the order of $GL(s,q)$, then $f(x)$ has the structure outlined in Equation (1). Consider the case where, in addition, $s > t$. Since $t \geq su$ and u is non-negative, u must be zero. If G has r blocks of size s, then g acts to fix all of the blocks. Therefore, g is in the kernel of the homomorphism from G to S_r. We may now apply Smash to our element g.

If p, the order of g, is also the characteristic of F, then the characteristic polynomial of g is always $(x-1)^d$ and offers no new information. Hence, we formulate a different test.

Let A be the matrix $(g-1)^{(p-1)}$. What can we say about the rank of A? As before, with respect to an appropriate basis, g acts on a single p-cycle of blocks of size s as s copies of a $p \times p$ permutation matrix of order p. If P is an arbitrary $p \times p$ permutation matrix of order p, then the rank of $(P - I_p)^{(p-1)}$ is exactly one, where I_p is the $p \times p$ identity matrix. More generally, $(P - I_p)^h$ has rank exactly $p - h$ for $1 \leq h \leq p$. For each p-cycle, there are s such permutation matrices and, hence, each p-cycle contributes s to the rank of A. Hence, the rank of A is $su + c$, for some $u, c \geq 0$.

This observation permits us to formulate the following test.

1. Compute the rank, t, of $(g-1)^{(p-1)}$. We call t the *free-rank* of g.

2. Find the smallest h where the rank of $(g-1)^h$ is equal to $t(p-h)$. Since the equality holds for $h = p-1$, such an h exists. We call h the *power-rank* of g.

3. First, consider the case where $s > t$. Since $t \geq su$, where u is non-negative, g must fix all blocks and we may supply the element to Smash.

4. Now, consider the case where s does not divide t. Since $t = su + c$ where both u and c are non-negative, and s does not divide t, it must be the case that $c > 0$. Therefore, there is a contribution to the rank of A from the fixed blocks. The action of g on the fixed blocks is represented by $s \times s$ submatrices. If y is an $s \times s$ matrix of order p in $GL(s, q)$, where $q = p^e$, then y is conjugate to an upper triangular matrix and $(y-1)^s$ has rank zero. If $s < p$, then clearly $(y-1)^{(p-1)}$ is the zero matrix and can make no contribution to the rank of A. Hence if $s < p$, then A must have rank su and we can rule out the existence of blocks of size s.

5. Finally, consider the case where $s < h$. By definition, h is the smallest integer such that $(y_i - 1)^h = 0$ for all y_i, where y_i is an $s \times s$ matrix representing the action of g on one of its fixed blocks. But $(y_i - 1)^s$ is the zero matrix, giving a contradiction if $s < h$. Hence, we can rule out the existence of blocks of size s.

We now describe a similar test for elements of prime-power order. Let g be an element of order p^n where $n > 1$. Let g have projective order p^m, where g^{p^m} is a scalar matrix in the element k of F.

If p differs from the characteristic of F, we find the largest power, t, of $x^{p^m} - k$ which divides f; otherwise, we compute the rank, t, of the matrix $(g-k)^{(p^m-1)}$. In each case, we call t the *free-rank* of g.

If $s > t$, then g cannot contain a cycle of length p^m in its action on blocks. Hence, $g^{p^{m-1}}$ must fix all blocks and we may supply this element to Smash.

In all cases, if Smash does not find a system of imprimitivity, then we can rule out the existence of all possible block sizes $s > t$, where t is the free-rank of the supplied element. Our call to Smash in Section 3 precludes the possibility that we now discover that G is semilinear.

4.3 Elements of composite order

Let g be an element of projective order o, whose prime factorisation involves distinct primes $p_1, p_2, \ldots, p_k$, where $k > 1$.

Can o be the order of an element of S_r? If not, then $\langle g \rangle$ cannot act faithfully on r blocks. Let p run over the distinct primes which divide o; then one of $g^{(o/p)}$ must fix all blocks. We supply each of these elements to Smash in turn.

In practice, we choose the element of projective order o which fails the membership test for the largest possible r.

In our earlier tests, which were described above, we applied a stronger version of this test to elements of prime-power projective order.

5 The block-stabiliser strategy

There are examples in which none of our existing tests is capable of deciding primitivity. In addition, SMASH can only find blocks of imprimitivity when there is some non-scalar matrix which fixes all of the blocks. We now describe a test which can find a block system when G acts faithfully as a permutation group on the blocks.

One deficiency of the tests in Section 4 is that they only consider the action of cyclic subgroups. The primary problem with this approach is that there exist primitive matrix groups in which every cyclic subgroup has a system of imprimitivity of the same block size, and also imprimitive groups in which every cyclic subgroup has so many systems of imprimitivity of the appropriate size that we need more information to find the block system.

The strategy described in this section uses subgroups of G that may be generated by more than one element. Unfortunately, it is rather slow. When it runs smoothly, it runs in time $O(d^5)$, but we are not able to prove conclusively that it will work in polynomial time in all cases. We have yet to encounter an example in which it fails completely, however.

The strategy is applied separately to each divisor s of d that has not been ruled out already. The remaining possible block sizes are processed in order of decreasing size. Hence, we may assume that the permutation action of G on the block system that we are seeking is primitive – otherwise, there would be another action with larger block size, which we would have already found.

5.1 Theory and outline of the strategy

Suppose that G does act imprimitively on V with blocks of size s, and let H be the stabiliser of one such block, W. Our strategy attempts to find H and W, or to establish that the assumption is false. If W exists, then V is isomorphic to the induced module W^G, where W is regarded as an FH-module. Thus, W must be irreducible as an FH-module, since otherwise V would not be irreducible as an FG-module. From Huppert ([7], Chapter V, Satz 16.6), we have $\mathrm{Hom}_{FG}(W^G, V) \cong \mathrm{Hom}_{FH}(W, V)$. (This module-theoretic generalisation by Nakayama of the Frobenius Reciprocity Theorem is valid over all fields F.) Since we are assuming throughout that V is an absolutely irreducible FG-module, $\mathrm{Hom}_{FG}(W^G, V)$ has dimension 1 over F. It follows that the only FH-submodule of V that is isomorphic to W is W itself. (We are grateful to L.G. Kovács for a helpful discussion on this argument.)

This suggests that we try to construct the stabiliser, H, of a fixed but unknown block, W, of size s. If we succeed in constructing H, then we can find W by first applying the MEATAXE algorithm to the action of H on V, and then, for each FH-composition factor V_i of dimension s, calculating $\mathrm{Hom}_{FH}(V_i, V)$. If $\mathrm{Hom}_{FH}(V_i, V)$ has dimension one, then W is the unique image in V of every nonzero homomorphism, and we can find the block system by applying MINBLOCKS to this image.

Since we assume that the permutation action of G on the blocks is primitive, H must be a maximal subgroup of G of index r. We try to construct H by working up a chain of subgroups, starting with a cyclic subgroup and then adjoining new generators. At some point in our construction, we may be able to decide that no such H exists, and thereby conclude that G does not preserve a block system with block size s.

More precisely, the algorithm iterates over a main loop. At the beginning of each iteration, we have a sequence $\mathcal{S}$ of subgroups of G, which are candidates for being subgroups of H. We start with $\mathcal{S} = \{\langle w \rangle\}$, where w is an element that must fix some block W of size s, if such a block system exists. The principal step is to find a collection $\{y_l\}$ of elements of G with the property that at least one of the y_l must fix the same block W. We then apply the MEATAXE to each of the subgroups $\langle K, y_l \rangle$, for all $K \in \mathcal{S}$. Usually, many of these subgroups will act irreducibly on V and can be discarded immediately. For those that act reducibly, we carry out more precise tests, during which we either find a block system, or we try to prove that the subgroup cannot lie in H. If there are subgroups $\langle K, y_l \rangle$ remaining for which we do not succeed in either of these aims, then we replace $\mathcal{S}$ by the sequence of such subgroups and begin a new iteration of the main loop. Otherwise we terminate.

In the following three subsections, we discuss how we choose both the initial element w and the collection $\{y_l\}$, and describe the tests we apply to the reducible subgroups $\langle K, y_l \rangle$. In the final subsection, we comment on the complexity of the whole process. We assume throughout that we have chosen particular values of the block size s and the number of blocks r, where $rs = d$.

5.2 The choice of w

Our first problem is how to choose the initial element w, which must be guaranteed to fix at least one block. For this, we make use of the powers of the random elements of G that we have accumulated during the earlier tests.

If there is an element g of prime-power projective order p^a, and p^b does not divide r for some $1 \leq b \leq a$, then we can clearly choose w to be $g^{p^{a-b+1}}$. For example, if $r = 12$ and g has projective order 9, then g^3 must fix a block. We choose p^{a-b+1} to be as large as possible, because we want the chain of subgroups to be as short as possible. In less obvious cases, we may be able to deduce that some power of g fixes a block by considering the characteristic polynomial of g, as described in Section 4.2. More precisely, assume g has prime-power projective order p^a and g^{p^a} is the scalar matrix kI_d for some $k \in F$; if the characteristic polynomial of g is not a power of $x^{p^a} - k$, then g cannot act on blocks with all of its orbits of length p^a, and so $g^{p^{a-1}}$ must fix a block. We can therefore choose w to be $g^{p^{a-1}}$.

If no suitable element can be found among our existing collection, then we compute the commutators of some pairs of the existing elements and test whether one of these commutators or a power of one fixes a block; this has worked in several examples, particularly when H is soluble.

If none of these attempts succeeds, we have no option but to put $w = 1$; in practice, we have never known this to happen.

5.3 The choice of the elements y_l

The next task is to choose the collection $\{y_l\}$. For this, we first find an element z of prime projective order p, where p is as large as possible.

We must first exclude the possibility that z fixes all of the r blocks: if p divides r, we do this by applying SMASH to z. If we conclude that z does not fix all r blocks, we know that z has between 1 and r/p cycles of length p in its action on the blocks. Let

$t = \mathrm{Int}(r/p) + 1$, and choose random elements $h_1, \ldots, h_t$ of G. Assume first that, in our putative block system, W is mapped to a block in a p-cycle of z by each element h_i. Then there exist h_{i_1} and h_{i_2}, with $i_1 \neq i_2$, which map W into the same p-cycle of z. Thus, at least one of the elements $h_i z^k h_j^{-1}$, for $0 \leq k < p$ and $1 \leq i < j \leq t$, must fix W. On the other hand, if some h_i maps W to a fixed point of z on blocks, then $h_i z h_i^{-1}$ must fix W. We therefore choose $\{y_l\}$ to be the set of all of these elements, $h_i z^k h_j^{-1}$ and $h_i z h_i^{-1}$. Note that there are $pt(t-1)/2 + t \leq (r+2)(r+p)/2p$ such elements, and so the larger p is, the smaller the number of y_l. In some situations, for example, when $r = p$, or when p divides r and does not divide the order of $GL(s,q)$ and the characteristic polynomial of z is a power of $(x^p - 1)$, we can easily deduce that z must act fixed-point-freely on the blocks, and so we need not include the conjugates $h_i z h_i^{-1}$.

In practice, we do not choose all of the h_i at once. Assuming that $h_1, \ldots, h_{u-1}$ have already been chosen for some $u \leq t$, we choose a random element h_u, and then calculate those elements y_l that have the form $h_i z^k h_u^{-1}$ for $1 \leq i < u$ and $h_u z h_u^{-1}$. We then process the subgroups $\langle K, y_l \rangle$ for $K \in \mathcal{S}$ and for these y_l. If this results in too many subgroups remaining unresolved, we immediately choose a new element h_u. We do this also if we are unable to distinguish (in terms of composition factors or minimal submodules) between V as an FK-module and V as an $F\langle K, y_l \rangle$-module for some y_l, because this could mean that $y_l \in K$, which is clearly undesirable.

5.4 Processing the subgroups $\langle K, y_l \rangle$

Finally, we describe in more detail how we process the subgroups $\langle K, y_l \rangle$. Let L be one of these subgroups. Suppose that, after applying the MEATAXE, we find that, as an FL-module, V has a_i composition factors of dimension d_i, where a_i and d_i are positive integers for $1 \leq i \leq n$, for some n, and $d_1 < d_2 < \ldots < d_n$. If L fixes a block W of dimension s, then W is an FL-submodule of V, and so there must be integers b_k for $1 \leq k \leq n$ with $0 \leq b_k \leq a_k$ for all k, such that $b_1 d_1 + \ldots + b_n d_n = s$. So we first find all solutions $b_1, \ldots, b_n$ to this equation, and if there are none, we reject L immediately. Otherwise, we consider each such solution in turn. We have devised three simple and efficient tests, each of which might apply to a particular solution. If at least one of the tests applies, then we either find a block system (in which case we can terminate the whole process immediately), or we rule out that solution. If we do not find a block system, and none of these tests applies to some solution, then we have failed to resolve this subgroup L. When this happens, we apply a final "desperate" test, which is theoretically conclusive, to this subgroup. However, in practice, this test is very expensive and we sometimes choose not to resolve L using it.

The first two tests use the feature described in Section 2.3. Let U be a composition factor of V as an FL-module. Then we can find some FL-submodule V_U of V that has U as a composition factor with positive multiplicity m, say, and no proper submodule of V_U has U as a composition factor with the same multiplicity m. The first test applies when $b_k = a_k$ for some k. In this case, the block W (if it exists) must contain all FL-composition factors of V of dimension d_k, and so it must contain V_U for every composition factor U of dimension d_k. We can test this immediately, by applying MINBLOCKS to V_U. The second test applies when there is a k with $b_k > 0$ such that all FL-composition factors U of V of dimension d_k have multiplicity 1. As we saw in Section 2.3, V_U is unique for all such U in this case, and so W must contain V_U for at least one composition factor U of dimension d_k. We therefore apply MinBlocks to each such V_U. For either test, if we do not find a block system, then we can rule out the relevant solution.

The third test applies only when $d_k = s$ and $b_k = 1$ for some k. For each FL-composition factor U of V having dimension s, we compute $\mathrm{Hom}_{FL}(U, V)$. If $\mathrm{Hom}_{FL}(U, V)$ has dimension 1, then V has a unique minimal FL-submodule isomorphic to U (which we compute as the image of an element in $\mathrm{Hom}_{FL}(U, V)$), and we apply MinBlocks to this submodule. We have included this test because, if there is a block system and L is the full stabiliser of the block, then, as we saw in Section 5.1, a call to MinBlocks will succeed in finding the system. If no block system is found, and $\mathrm{Hom}_{FL}(U, V)$ has dimension greater than 1 for some composition factor U of dimension s, then the test is inconclusive.

For the final "desperate" test, we consider the set of those degrees, d_k, which have positive coefficient, b_k, in some remaining solution. We now compute $\mathrm{Hom}_{FL}(U, V)$ for all FL-composition factors U of V having dimension d_k. The idea is that we then compute all minimal submodules of V that are isomorphic to some such U of dimension d_k and apply MinBlocks to each of them. In principle, this test either finds a block system, or it conclusively rules out the subgroup L. However, the number of such submodules can sometimes be impracticably large. Hence, we choose some positive integer Max and compute at most Max minimal submodules isomorphic to a particular U. The value of Max can be increased with each iteration of the main algorithm. If the upper limit is exceeded for U, it does not always imply that the test fails. For example, if there is just one remaining solution, which involves composition factors of different degrees, then it is only necessary to compute the minimal submodules for one of these degrees. More generally, suppose that we can find a subset D of the d_k with the property that each unresolved solution involves at least one factor of dimension c for some $c \in D$. Then it suffices to compute all minimal submodules for all composition factors of V of dimension c, for all $c \in D$. There may be more than one choice of the subset D. If so, we choose that subset which minimises the number of minimal submodules that have to be calculated. This number can be predicted in advance by calculating the dimensions of $\mathrm{Hom}_{FL}(U, V)$.

In practice, the block-stabiliser test will not succeed in the following situation: G is primitive, but one of the subgroups L has the property that we cannot resolve it by applying one of our three fast tests to it, the "desperate" strategy is too slow, and whenever we augment L to produce the subgroups $\langle L, y_l\rangle$, one of these is L again.

5.5 Some remarks on complexity

As we saw in Section 5.3, each of the collections $\{y_l\}$ has about $r^2/2p = d^2/2ps^2$ elements, where p is the order of the element z. We have to apply the Meataxe, which runs in time $O(d^3)$, to each of the subgroups $\langle K, y_l\rangle$. Thus, provided that we do not have to iterate the principal loop too many times, and the sequences $\mathcal{S}$ do not grow too large, the whole process runs in time $O(d^5)$. While this is slower than we would wish, it is tolerable for dimensions up to about 100. In a typical straightforward application, we might have to go through two cycles with $\mathcal{S}$ of size one in the first and of size at most 10 in the second, and so this estimate is quite an accurate guideline in many cases.

We are not able to prove complexity results formally, however, and we have encountered isolated examples which behaved badly using our earlier implementations. We have tried to identify the situations where things might go wrong, and then attempted to find remedies. One danger is that the chain of subgroups going up to H could turn out to be very long (possibly of order d). Fortunately, this does not seem to be common. Our simpler tests primarily fail for examples which seem to be fairly close to being simple groups, and their maximal subgroups usually have a small number of generators.

A more serious danger, and one that we have encountered, is that there is a "rogue" subgroup of G which is not the stabiliser of a block, but which acts reducibly on V in such a way that we cannot prove that it is not contained in the stabiliser of a block. Then the iteration process can get stuck inside this subgroup. We have observed, in practice, that by being careful in our choice of the random elements h_i, as we described in Section 5.3, we can prevent the ascending chain of subgroups becoming constant. Since we also steadily increase the limit MAX defined in Section 5.4, we can hope to rule out these cases eventually. These two measures have sufficed in all of the examples considered so far.

6 Implementing the algorithm

An implementation of the algorithm was developed in GAP and is publicly available. Our implementation takes as input a generating set for a matrix group and reports one of the possible outcomes of the algorithm. Here, we discuss some of the practical considerations which arose in developing an implementation.

A detailed discussion of the method used to select random elements is provided in Holt & Rees [5]; their method is based on an idea of Leedham-Green. Essentially, a certain amount of preprocessing is first carried out; this allows the selection of a new random element for the cost of one matrix multiplication.

Most matrix operations carried out, including the characteristic polynomial calculation, cost $O(d^3)$. Both the order and projective order of an individual matrix can be found in $O(d^3 \log q)$ in practice, following an algorithm devised and implemented by Celler and Leedham-Green.

In the characteristic polynomial test for elements of prime order, we decide whether p divides the order of $GL(s,q)$ by computing the smallest integer m such that $q^m - 1$ is divisible by p and then checking whether $s < m$.

Since the order test described in Section 4.1 is not expensive and is frequently highly effective, about 20 random elements are first selected and their orders computed. Each random element and its order is stored. The elements of prime-power order used in the tests of Section 4.2 are constructed by taking powers of these elements.

In practice, SMASH is an expensive part of the computation. Hence, we seek to minimise the number of applications of this procedure. As a consequence, all potential elements of the kernel of the homomorphism from G to S_r found in the tests of Section 4.2 are stored in a SMASH queue, together with the values of their free-ranks. When these tests have been applied to all of the elements of prime-power order generated as powers of the random elements selected, we then choose the element of smallest free-rank, t, from the SMASH queue and supply this element as input to SMASH. Now, either we find a block system for G and hence terminate the test, or we can rule out all $s > t$.

Recall, from Section 5.2, that we try to find a non-trivial element, w, which fixes at least one block: if our initial collection of random elements does not provide a suitable candidate, we select about 10 new random elements; if we do not find a suitable element among the powers of these, we then compute about 10 commutators of pairs of elements; we iterate both of these steps at most three times before abandoning our search and choosing w to be the identity.

7 Some sample applications

Below we report in some detail on the application of our algorithm to a range of test cases. These demonstrate that every one of our tests is used in order to settle existing examples conclusively. We use the notation of the Atlas [3] to identify each group.

- A 20-dimensional representation of A_7 over the field of 2 elements. Here the potential block sizes are $1, 2, 4, 5, 10$. Elements of order 3 and 7 which do not have characteristic polynomials of the form described in Equation (2) eliminate 1 and 2 respectively. The element of order 7 has free-rank 2; it is supplied to Smash and this call eliminates the remaining possible block sizes.

- A 24-dimensional representation of $2Co_1$ over the field of 3 elements. Here the potential block sizes are $1, 2, 3, 4, 6, 12$. One element of order 3 and power-rank 2, and another of order 3 and free-rank 5, eliminate 1 and 2, respectively. An element of order 9 has free-rank 1; its cube is supplied to Smash and this call eliminates the remainder.

- A 25-dimensional representation of $A_5 \times A_5$ over the field of 7 elements. Here the potential block sizes are 1 and 5. An element of order 10 in the composite order test generates a call to Smash which finds a tensor product decomposition, where each factor has dimension 5. A recursive application of the primitivity algorithm finds blocks of size 5.

- A 28-dimensional representation of A_8 over the field of 11 elements. Here the potential block sizes are $1, 2, 4, 7, 14$. The block-stabiliser test is used to eliminate each of 1 and 2: w has order 3, z has order 7, no conjugates of the y_is are needed, but the "desperate" test is called. Another call to the block-stabiliser test – this time, with an element w of order 4 – eliminates 4. But w has free-rank 4; its square is supplied to Smash and this call eliminates the remainder.

- A 30-dimensional representation of $U_4(2)$ over the field of 49 elements. Here the potential block sizes are $1, 2, 3, 5, 6, 10, 15$. One element of order 9 and free-rank 3 eliminates all values which are at least 5. Invocations of the block-stabiliser test, with w of order 4 and 9, z of order 5, and one call to the "desperate" test eliminates each of 2 and 3, respectively. Another invocation of the block-stabiliser test, this time with w of order 3 and z of order 5, requires a second pass with a sequence $\mathcal{S}$ of length 4 before a call to the "desperate" test eliminates 1.

- A 32-dimensional representation of $L_2(31)$ over the field of 16 elements. Here the potential block sizes are $1, 2, 4, 8, 16$. The order test with an element of order 31 eliminates 2 and 4. Since this element has free-rank 1, it eliminates all possible block sizes except 1. The block-stabiliser test, with each of w and z having order 31, finds two block systems.

- A 50-dimensional representation of $He2$ over the field of 7 elements. Here the potential block sizes are $1, 2, 5, 10, 25$. The free-rank, 6, and power-rank, 5, of an element of order 7 eliminates 1, 2, and 5; the element is supplied to Smash and this call eliminates the remainder.

- A 55-dimensional representation of $M_{11} \wr M_{11}$ over the field of 7 elements. Here the potential block sizes are $1, 5, 11$. An element of order 8 has free-rank 0; its fourth power is supplied to Smash which finds 11 blocks of size 5.

- A 90-dimensional representation of $3O'N2$ over the field of 7 elements. The call to Smash which seeks to decide whether the group is semilinear finds a block system containing two blocks of size 45.

- A 111-dimensional representation of Ly over the field of 5 elements. Here the potential block sizes are $1, 3, 37$. An element of order 5, which does not have a characteristic polynomial of the form described in Equation (2), eliminates both 1 and 3. The element has free-rank 15, and a call to Smash eliminates 37.

Recall that the algorithm may report that a group is semilinear and consequently it is not able to decide whether or not the group is imprimitive. We now present a simple example to illustrate this possible outcome.

Consider a 2-dimensional representation of the cyclic group of order 3 over $GF(2)$. Take the wreath product of this group with S_3 to get a 6-dimensional imprimitive representation, which is not absolutely irreducible. Adjoin a generator of order 2 which fixes all 3 blocks and inverts all of the 3-elements in the base group of the wreath product. The group clearly remains imprimitive, and the involution makes it both absolutely irreducible and semilinear. The reduction test of Section 3 discovers that the group is semilinear and the algorithm terminates.

8 Implementation performance

Here, we present a range of examples to provide a guide to the performance of our GAP implementation of the algorithm. Where possible, we report tests for a range of representations of the same group – of different dimensions over the same field, and of the same dimension over different fields – to give some indication of the sensitivity of the algorithm to changes in dimension and field.

All computations were carried out using GAP Version 3.2 on a SPARC Station 10/51, and all CPU times are given in seconds. Twenty random elements of each group were selected for the order test. In Tables 1 and 2, for each group, we list its Atlas name, report its dimension, the finite field it is defined over, whether it is imprimitive or not, and the CPU time taken. We indicate that a group is primitive or imprimitive by listing “P” or “I”, respectively, in the Status column of each table.

Since the algorithm has a random component, the times listed should be viewed only as a general guideline. In an attempt to provide a realistic guide to performance, we report the CPU time averaged over three consecutive executions.

Short [12] determined the primitive soluble permutation groups of degree less than 256 and hence constructed a list of soluble subgroups of small dimensional general linear groups. The application of our implementation to each of these groups took at most one second of CPU time.

Group	Dimension	Field	Status	Time
A_5	5	7	I	5
$A_5 \times A_5$	25	7	I	62
A_7	20	2	P	14
A_8	20	11	P	15
A_8	28	11	P	552
A_8	64	11	P	3293
A_9	28	49	P	360
$2A_{11}$	55	5	P	107
$2Co_1$	24	3	P	15
Co_3	22	3	P	13
$F_4(2)$	26	2	P	18
Fi_{23}	253	3	P	10291
$He2$	50	7	P	80
$He2$	102	2	I	169
J_1	7	11	P	3
J_1	14	11	P	13
J_1	27	11	P	12
J_2	36	3	P	130
J_2	42	3	P	1571
$2J_2$	12	3	P	7
$3J_3$	18	2	P	11
$3J_3$	36	2	P	36
$3J_3$	80	2	P	218
J_4	112	2	P	318
$L_2(13)$	14	7	I	18
$L_2(17)$	18	41	I	35
$L_2(31)$	32	16	I	116
$L_2(81)$	82	41	I	1706
$L_3(3)$	26	2	I	46
$L_3(4)$	63	5	I	613
$L_3(4)$	63	11	I	739

Table 1: Performance of implementation for a sample of groups

Group	Dimension	Field	Status	Time
$L_3(5)$	124	2	I	1975
$L_3(5)$	124	3	I	1305
$L_3(5)$	124	31	I	2064
Ly	111	5	P	304
M_{11}	24	3	P	10
M_{11}	44	2	P	162
M_{11}	44	7	P	224
M_{11}	55	7	I	344
$M_{11} \wr M_{11}$	55	3	I	93
M_{12}	55	7	P	2903
M_{12}	120	17	I	1062
M_{22}	21	7	P	10
M_{22}	30	2	P	15
M_{22}	34	2	P	27
M_{22}	54	7	P	60
M_{22}	154	7	P	10128
$3M_{22}$	12	2	P	6
$3McL$	21	5	P	8
$3McL$	90	5	P	302
$(3^2 : 4 \times A_6) \cdot 2$	18	7	P	93
$(3^2 : 4 \times A_6) \cdot 2$	27	7	P	28
$3O'N2$	90	7	I	86
Ru	28	2	P	20
Ru	28	5	P	14
Ru	28	17	P	41
Suz	12	3	P	5
Suz	12	4	P	4
$Sz(8)$	65	29	I	731
Th	248	2	P	48265
$U_4(2)$	14	2	P	4
$U_4(2)$	30	49	P	3907
$U_4(2)$	58	5	P	94
$U_4(2)$	64	2	P	649
$U_4(2)$	81	11	I	905

Table 2: Performance of implementation for a sample of groups

References

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

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

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

[4] Derek F. Holt, C.R. Leedham-Green, E.A. O'Brien and Sarah Rees, "Computing matrix group decompositions with respect to a normal subgroup". Preprint.

[5] Derek F. Holt and Sarah Rees, "An implementation of the Neumann–Praeger algorithm for the recognition of special linear groups", *J. Experiment. Math.*, **1**, 237–242, 1992.

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

[7] B. Huppert, *Endliche Gruppen I.* Grundlehren Math. Wiss. **134**, 1967. Springer-Verlag, Berlin, Heidelberg, New York.

[8] Peter M. Neumann and Cheryl E. Praeger, "A recognition algorithm for special linear groups", *Proc. London Math. Soc.*, **65**, 555–603, 1992.

[9] R.A. Parker, "The computer calculation of modular characters (the Meat-Axe)", M.D. Atkinson (Ed.), *Computational Group Theory*, (Durham, 1982), pp. 267–274, 1984. Academic Press, London, New York.

[10] Michael Ringe, *The C* MeatAxe. Lehrstuhl D für Mathematik, RWTH, Aachen, 1992.

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

[12] M.W. Short, *The Primitive Soluble Permutation Groups of Degree less than* 256, Lecture Notes in Math., **1519**, 1992. Springer-Verlag, Berlin, Heidelberg, New York.

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