An Implementation of the Neumann–Praeger Algorithm for the Recognition of Special Linear Groups

Derek F. Holt and Sarah Rees

CONTENTS

We report on our implementation of an algorithm due to Neumann and Praeger for deciding whether or not a matrix group over a finite field contains the special linear group. This is a Monte Carlo algorithm, and thus has a small but precise probability of returning the wrong answer; this probability can be specified in advance by the user. The algorithm requires the selection of random elements from the group, and the most important problem that arose in the implementation was to find a satisfactory procedure for making this selection.

1. INTRODUCTION

The purpose of this article is to report on our implementation of the algorithm described in [Neumann and Praeger 1992] for solving the following problem. Let q be a prime power, d a positive integer, and $\mathrm{GL}(d, q)$ the linear group of nonsingular $d \times d$ matrices over the finite field F_q of order q. Given a finite subset X of $\mathrm{GL}(d, q)$, does the group G generated by X contain the special linear group $\mathrm{SL}(d, q)$?

This implementation has been carried out in the GAP system developed by Martin Schönert and others at Aachen [Schönert et al. 1992]. We have followed the algorithm described in [Neumann and Praeger 1992] fairly closely, and so we only need to report on one or two minor deviations, and on those parts of the process that were not described precisely there. The implementation is practical for values of d up to at least 60 or 70, and it works reasonably well for all F_q that are currently known to GAP, namely those with $q \leq 2^{16} = 65536$.

In Section 2 we outline the main steps of the algorithm and present some running times for our implementation. We emphasise that we are dealing

1058-6458/92 $0.50 per page

with a Monte Carlo algorithm, which means that there is a number ε (which the user can choose in advance) such that there is a probability of at most ε that the process will give an incorrect answer. In this particular case, only false negatives can occur. In other words, the answer "yes" can always be relied upon, but the answer "no" may very occasionally be incorrect.

The theoretical analysis of the process requires that we make a moderately large (up to a few hundred) number of independent choices of random elements of the group G. A Monte Carlo algorithm to generate (pseudo-)random elements of G is described in [Babai 1991]. Unfortunately, this does not seem to be practical for us, on account of the large number of matrix multiplications that it requires. We have therefore been forced to adopt a procedure that does not attempt to cover the whole group with uniform probability, but merely to choose elements in such a way that they have the correct probability of having the desired properties. We must then rely on heuristical and statistical evidence to support the method that we have adopted. These matters are discussed in more detail in Section 3.

Finally, since the description in [Neumann and Praeger 1992] is not quite precise for $d \leq 4$, we fill in the details of these cases in Section 4.

According to a result of Aschbacher [1984], any matrix group over a finite field lies in one of nine classes of matrix groups, and one of these classes consists of groups that contain the special linear group. We hope that the Neumann-Praeger algorithm will be the first of a series of algorithms that seek to recognise whether a particular group lies in one or other of these classes.

In fact, there has already been a suggested improvement to the Neumann–Praeger algorithm itself by Charles Leedham-Green, which has been implemented in GAP by Frank Celler. This currently lacks a precise theoretical probabilistic analysis, but statistical results suggest that it involves looking at far fewer random elements of the group. What seems to be clear, however, is that all algorithms of this type are likely to depend strongly on the selection of random elements from the group, and so an investigation of this process is essential.

2. OUTLINE OF THE ALGORITHM AND TIMINGS

We begin with a very brief outline of the Neumann–Praeger algorithm; readers should consult [Neumann and Praeger 1992] for more details. An element g of $\mathrm{GL}(d,q)$ is called *irreducible* if it acts irreducibly on the underlying d-dimensional vector space V over F_q, and it is called *nearly irreducible* if it fixes and acts irreducibly on a $(d-1)$-dimensional subspace of V. The element g is called *primitive irreducible* if it is irreducible and its order $|g|$ is divisible by some prime p that divides q^d-1 but does not divide q^e-1 for any positive integer $e<d$. Similarly, g is called *primitive nearly irreducible* if it is nearly irreducible and $|g|$ is divisible by a prime dividing $q^{d-1}-1$ but not dividing q^e-1 for any $e<d-1$. Finally, g is called *ample* if no conjugate of any element gz, with z a scalar matrix, lies in $\mathrm{GL}(d,r)$ for any proper subfield F_r of F_q.

It is shown in [Neumann and Praeger 1992] that, with a few precisely described exceptions, any subgroup of $\mathrm{GL}(d,q)$ that contains an ample element, a primitive irreducible element, and a primitive nearly irreducible element must contain the whole of $\mathrm{SL}(d,q)$. It is also shown that, if G is a subgroup of $\mathrm{GL}(d,q)$ that contains $\mathrm{SL}(d,q)$, the proportion of ample primitive irreducible elements of G is at least $1/(d+1)$, except when $d<3$ or $(d,q)=(6,2)$, and the proportion of ample primitive nearly irreducible elements is at least $1/d$, except when $d<4$ or $(d,q)=(7,2)$.

The algorithm proceeds as follows, with slight modifications for small d and the exceptional cases $(d,q)=(6,2)$ and $(7,2)$. Choose a sample S of n random elements of the given group G, where n is such that, if G does contain $\mathrm{SL}(d,q)$, the probability that S does not contain both an ample primitive irreducible element and an ample primitive nearly irreducible element is less than the chosen error probability ε. (It is easy to calculate n from ε and d; if $\varepsilon=\frac{1}{100}$, for example, $n=\frac{11}{2}d$ is sufficient.) Check the elements in the sample for ampleness, primitive irreducibility and primitive near-irreducibility. If the sample does not contain an ample element, or if it does not contain a primitive irreducible element, or if it does not contain a primitive nearly irreducible element, return the answer "no". If it does contain each of these three elements, check for the known exceptional examples, and answer accordingly.

Now, from our implementation, it turns out that the proportion of computation time taken by tests for primitivity, ampleness, and dealing with the exceptional cases is very small. This is essentially because they only have to be done once or twice, rather than for each individual element of the sample S. The test for primitivity is described in [Neumann and Praeger 1992, p. 578–579] and consists of raising the element to an appropriate high power and checking if the result is the identity. Since raising matrices to high powers is fairly efficient, this is not too time-consuming. The test for ampleness is described in [Neumann and Praeger 1992, pp. 578], and involves some simple tests on the coefficients of the characteristic polynomial of the element. The exceptional cases are dealt with by individual tests; typically, these involve some short orbit calculations. For example, in the case $n = 11, q = 2$, the exceptional case in which G is the Mathieu group M_{24} is recognised by the fact that an eigenvector for the nearly irreducible element has precisely 1288 translates under the action of the group.

The bulk of the time is taken up with calculating the random elements and testing them for irreducibility and near-irreducibility. We shall discuss the first of these operations in the next section, but here we simply note that the procedure we have adopted is to first enlarge the generating set by using a moderate number (about 10) of relatively long words (length about 30) in the original generators. This takes a certain amount of time, but only needs to be done once. We then continually replace a randomly chosen generator by its product with another, and use these products as the elements of S.

For the testing of the elements g in S for irreducibility and near-irreducibility, Neumann and Praeger suggest calculating the characteristic polynomial f of g, which is irreducible if and only if g is irreducible, and has an irreducible factor of degree $d-1$ if and only if g is nearly irreducible. There was already a GAP procedure available, written by Frank Celler, for calculating the minimal polynomial, and it was a simple matter to adapt it to calculate the characteristic polynomial instead. In fact, following a suggestion of the referee, we can take a short cut by interrupting this procedure whenever an invariant subspace for g of dimension lying strictly between 1 and $d-1$ is found, since in that situation g cannot possibly be irreducible or nearly irreducible. However, it turns out that this shortcut helps significantly only for very small fields and relatively large degrees. For large fields it is extremely rare to find such a subspace: for example, in the symplectic groups $\mathrm{Sp}(60, q)$ with $q = 2, 3, 5$ and 11, such a subspace was found for about $36\%, 15\%, 6\%$ and 1% of the elements, respectively.

There were already some GAP procedures available for factorising polynomials over finite fields, and we were able to adapt them to test only for irreducibility and near-irreducibility. These procedures work by first finding the linear factors of the polynomial f, then finding the quadratic factors, and so on. Since the existence of any proper factor will rule out irreducibility (and usually also near-irreducibility), this means that we can often stop long before f has been factorised completely. This process will therefore take longest when f really is irreducible, but that does not matter, since it is an irreducible element that we are seeking.

Table 1 shows some running times for our implementation, obtained on a Solbourne 5/600, a workstation similar in performance to a Sun Sparcstation 2. Since this algorithm only runs to completion when G does not contain $\mathrm{SL}(d, q)$, all of the rows except the last represent examples with a negative answer. In the last example, the time given is an average over 500 runs. Times can vary by at least 10% on different runs of the same example; this applies particularly to t_{red}, which depends strongly on the polynomials involved.

We have tried to choose examples that demonstrate the effects of changing the degree, the field and the group. Thus, the basic example, of which the others are variations, is $G = \mathrm{Sp}(40, 17^2)$. It appears to be t_{minp} that is most affected by the degree, whereas t_{red} is most affected by changing the field or the group. Concerning the field, it seems to be increasing the characteristic rather than increasing the size that has the worse effect; it is not clear why this should be the case, but it stems from the basic field and matrix operations within GAP. The two cyclic groups C_1 and C_2 were chosen as examples that we expected to be particularly quick and particularly slow, respectively. Thus C_1 (of order $17^2 - 1$) consists entirely of diagonal matrices, and so all of the operations involved are as quick as they ever could be, whereas C_2 (of order $17^{20} - 1$)

G	d	q	n	$t_{\rm pre}$	$t_{\rm ran}$	$t_{\rm minp}$	$t_{\rm red}$	$t_{\rm tot}$
$\mathrm{Sp}(20, 17^2)$	20	17^2	110	13	3	18	35	71
$\mathrm{Sp}(40, 17^2)$	40	17^2	215	82	47	188	178	496
$\mathrm{Sp}(60, 17^2)$	60	17^2	320	254	220	698	916	2119
$\mathrm{Sp}(40, 2)$	40	2	215	14	9	63	22	109
$\mathrm{Sp}(40, 2^{16})$	40	2^{16}	215	107	78	193	555	934
$\mathrm{Sp}(40, 1009)$	40	1009	215	88	56	168	339	652
$\mathrm{Sp}(40, 5003)$	40	5003	215	204	119	230	1289	1844
$\mathrm{Sp}(40, 10007)$	40	10007	215	354	197	315	1790	2658
C_1	40	17^2	215	10	5	58	39	114
C_2	40	17^2	215	36	19	80	1057	1193
$\mathrm{GL}(20, 17^2)$	20	17^2	30					35

TABLE 1. Timings, in seconds, for representative runs of our implementation of the Neumann–Praeger algorithm, on a Solbourne 5/600. The column labeled n gives the number of random elements considered; $t_{\rm tot}$ is the total CPU time for the run, broken down into the preprocessing time $t_{\rm pre}$ for extending the original set of generators, the time $t_{\rm ran}$ to calculate the random elements, the time $t_{\rm minp}$ to compute their minimal polynomials, and the time $t_{\rm red}$ to decide whether they are reducible or nearly irreducible. $\mathrm{Sp}(d, q)$ are symplectic groups, and C_1 and C_2 are particular cyclic groups (see text). The last row shows averages over 500 runs.

is generated by an element g of which the minimal polynomial is a product of two distinct irreducible factors of degree 20. This should be the worst possible situation for the reducibility test.

3. THE SELECTION OF RANDOM GROUP ELEMENTS

We have put a considerable amount of effort into trying to find a satisfactory method of choosing random elements from the given matrix group; this seems to be worthwhile, because such a method will be needed in the future for many other matrix group algorithms. The main problem is that, if the chosen procedure involves more than about five matrix multiplications for each random element, the total time $t_{\rm ran}$ taken for this part of the procedure becomes inordinately large in comparison with the time for the other parts. After much experimentation, the procedure that we eventually adopted involves a certain amount of preprocessing, but thereafter requires only a single matrix multiplication for each random element. This procedure, based on ideas of Charles Leedham-Green and Leonard Soicher, was suggested to us by the referee of the original version of this paper (in which a slightly different method was described).

We had first considered two procedures that attempted to choose every element in the group with roughly equal probability (which is of course what is necessary for a truly random process). The first is described in [Babai 1991] and is a very general method, applicable whenever we know an upper bound N on the group order and can multiply group elements. It yields elements that are guaranteed to be random, so long as a certain preprocessing algorithm succeeds; this happens with probability at least $1 - \varepsilon$, where ε can be chosen beforehand. Unfortunately, the method has a lengthy preprocessing phase involving $O(\log^5 N)$ group multiplications, after which each random element itself requires $O(\log N)$ multiplications. We have $\log N = d^2 \log q$ for matrix groups; since we are aiming to go up to at least $d = 60$ and $\log q = 16$, this process involves far too many operations.

The second suggestion is made by Neumann and Praeger themselves in their paper. It consists of choosing products of powers of the form

$$g_1^{m_1} g_2^{m_2} \dots g_l^{m_l},$$

where $l \le 2d$, the g_i are randomly chosen elements from the given set of generators of G, and the m_i are random integers in the range $1 \le m_i \le q^d$. The reason for considering powers is that they can be computed fairly efficiently and, provided that the generators do not all have small order, there are sufficiently many products of this form to make it heuristically plausible that they should cover the set of all group elements with equal probability. However, we timed this process with the group $\mathrm{GL}(40, 5^5)$ and found that calculating each such product in GAP required about 278 seconds on average and, since we would need to find up to 215 such products in dimension 40, this would still make this part of the complete procedure disproportionately slow in comparison with the rest.

We therefore decided to abandon the aim of selecting all group elements with equal probability. After all, we only need to select our sample in such a way that the probability of an element chosen being irreducible or nearly irreducible is about the same as the proportion of such elements in the whole group. The difficulty with this is that we can conceive of no method of justifying such a procedure theoretically or of estimating any small deviation between the actual and the expected probabilities. This is unfortunate, since it mars the otherwise impeccable mathematical analysis of the algorithm. However, we believe that we have extremely convincing statistical evidence to justify the procedure that we eventually chose.

Here we have a new problem. Any statistical argument will be based on experiments, and any experiment will use a particular set of generators. But we want our procedure to be equally valid for all generators, and we cannot possibly test all possible sets of generators; choosing random generators is definitely not helpful, since procedures like this are almost by definition likely to perform best on random collections of elements. It is rather the improbable generating sets that we need to worry about, whatever this may mean. In fact, we are going to rely on the slightly dubious assumption that the generators for $\mathrm{GL}(d,q)$ and $\mathrm{SL}(d,q)$ described in [Taylor 1987] are as improbable as any. The two generators of $\mathrm{GL}(d,q)$ are A and B, where A is a diagonal matrix with $A_{11} = w$ (a field generator) and $A_{ii} = 1$ for $i > 1$, and B is defined by $B_{11} = -1$, $B_{1d} = 1$, $B_{i,i-1} = -1$ for $2 \leq i \leq d$, and $B_{ij} = 0$ otherwise. They are both extremely sparse and A, at least, is highly improbable in the sense that the proportion of diagonalisable elements in the group is vanishingly small except for very small values of d.

We used the following statistical test to evaluate selection procedures. For a matrix A, let $h(A)$ be the highest degree of an irreducible factor of the minimal polynomial of A. Now, if G is any group with $\mathrm{SL}(d,q) \subseteq G$ and k is any integer with $k > \frac{1}{2}d$, arguments similar to those in [Neumann and Praeger 1992] show that, if $d > 2$ and A is a random element of G, the probability that $h(A) = k$ lies between $1/k$ and $1/(k+1)$. In fact, unless q is very small, this probability lies much closer to $1/k$. We therefore used the proposed procedure to select a large sample of elements, computed $h(A)$ for each of these elements, and compared the expected frequencies with the observed frequencies, grouping all elements with $h(A) \leq \frac{1}{2}d$ into a single class. We could then carry out a χ^2 test for significant deviation.

We soon concluded that simply choosing words in the original generators is inadequate. Even with words of length 20, the value of χ^2 was outside of the 0.05 probability zones, whereas with length 50, a sample of size 1000 still contained 61 elements A with $h(A) = 1$, but the proportion of such elements in G is much smaller. When we started with two random elements as generators rather than the elements from [Taylor 1987], we still found that choosing random words of length 5 in them was completely inadequate, and words of length 10 was suspect. The solution was therefore to start with a much larger set of generators; this seemed to have the desired effect of eliminating or at least greatly reducing any biases resulting from the effects of particular generators.

The procedure we have finally employed is the following. If there are n generators to begin with, we introduce $\max(10, n)$ new generators, each of which is chosen as a random word of length about 30 in the existing generators. This is part of the preprocessing phase. We then perform the following process repeatedly. We choose two distinct generators x and y at random, and replace x by xy. We do this n^2 times (where n is now the current number of generators) as part of the preprocessing phase. Thereafter, we use the new generator xy as the required random element. It is quite plausible (and may even be provable) that the resulting elements will eventually cover the whole group with equal probability, although of course successively chosen elements will not be distributed uniformly amongst pairs of group elements.

We carried out our significance test with this procedure on a variety of examples with different values of d and q, and we never obtained a value of χ^2 outside of the 0.005 probability zone. (In fact, we suspect that we are doing rather more preprocessing than is necessary for our purposes.) We present one such set of results in Figure 1.

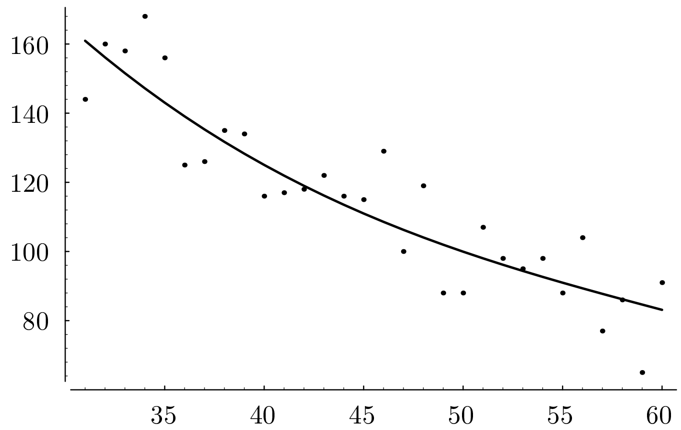

FIGURE 1. Expected frequency (line) and observed frequency (dots) for each value of $h(A)$ from 31 to 60, in a 5000-element sample in $\mathrm{GL}(60, 5^5)$. Expected and observed frequencies for $h(A) \leq 30$ were 1576 and 1609. The χ^2 for this data is 30.14, almost the same as the expected value of 30.

4. THE LOW-DIMENSIONAL CASES

The algorithm in [Neumann and Praeger 1992] is only specified precisely for dimensions $d > 4$, and a more general matrix group recognition procedure is described for $d \leq 4$. Since we preferred to make our program uniformly applicable in all dimensions, we shall quickly describe the necessary modifications necessary for the low-dimensional cases.

For $d = 4$, it follows from [Neumann and Praeger 1992, Theorem 2] that the only exceptional groups G containing primitive irreducible and nearly irreducible elements are those with $G/Z \cong A_7$, where Z denotes the subgroup of scalar matrices in G. Furthermore, this can only occur when $q = 2$ or $q \geq 23$. In the first case, we can simply calculate the order of G, which is 2520 if and only if $G \cong A_7$. In the second case, we consider the action of G on the set Ω of one-dimensional subspaces of V, and compute the length l of the orbit of $\langle v \rangle$ under G, where $\langle v \rangle$ is fixed by a primitive nearly irreducible element of G. Then $G/Z \cong A_7$ if and only if $l \leq 120$.

For $d = 3$, to maintain the validity of the probabilistic analysis [Neumann and Praeger 1992, Lemmas 2.5 and 2.6], we drop our requirement that the nearly irreducible element that we are seeking should be primitive. By considering the list in [Neumann and Praeger 1992, Proposition 8.2], we find that the exceptional groups are the semilinear groups $\Gamma\mathrm{L}(1, q^3)$ and groups with $G/Z \cong \mathrm{PSL}(2, 7)$. These are all among the list of exceptions for general d, so we already have procedures to recognise them.

For $d = 2$, we deal with the very small fields ($q \leq 5$) simply by calculating the order of G. For $q \geq 7$, it is easy to show that, if $\mathrm{SL}(2, q) \subseteq G$, at least $|G|/4$ of the elements g of G have the property that g is ample and g^2 is irreducible. We now seek an element with this property, and consider enough random elements to guarantee that failure occurs with probability at most ε. If we find such an element, the exceptional possibilities are $G \cong \Gamma\mathrm{L}(1, q^2)$ and $G/Z \cong A_4, S_4$ or A_5. The last three can be tested for by calculating the length l of the orbit of a subspace $\langle v \rangle$ under G, as above. If $l \leq 60$, we have one of the exceptions.

Finally, if $d = 1$, we return the answer "yes".

REFERENCES

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

[Babai 1991] L. Babai, "Local expansion of vertex-transitive graphs and random generation in groups", *Proc. 23rd ACM Symp. Theory of Computing, New Orleans* (1991), 164–174.

[Neumann and Praeger 1992] P. M. Neumann and C. E. Praeger, "A recognition algorithm for special linear groups", *Proc. London Math. Soc.* **65** (1992), 555–603.

[Schönert et al. 1992] M. Schönert et. al., *GAP: Groups, Algorithms, and Programming*, Lehrstuhl D für Mathematik, RWTH Aachen, Germany, 1992. Available, together with the GAP system, by anonymous ftp from the `/pub` directory on servers `samson.math.rwth-aachen.de` or `dimacs.rutgers.edu`.

[Taylor 1987] D. Taylor, "Pairs of generators for matrix groups", *The Cayley Bulletin* (Univ. of Sydney) **3** (1987), 76–85.

Derek F. Holt, Mathematics Institute, University of Warwick, Coventry CV4 7AL, Great Britain (dfh@maths.warwick.ac.uk)

Sarah Rees, Department of Mathematics and Statistics, Merz Court, The University, Newcastle-upon-Tyne NE1 7RU, Great Britain (sarah.rees@newcastle.ac.uk)

Received July 20, 1992; revised November 12; accepted November 16