GENERATING RANDOM ELEMENTS OF A FINITE GROUP

Frank Celler
Lehrstuhl D für Mathematik, RWTH, 52062 Aachen, Germany

Charles R. Leedham-Green
School of Mathematical Sciences, Queen Mary and Westfield College,
University of London, London E1 4NS, United Kingdom

Scott H. Murray
Department of Mathematics, University of Chicago, Chicago, IL 60637, USA

Alice C. Niemeyer
Mathematics Department, University of Western Australia, Nedlands,
WA 6009, Australia

E.A. O'Brien
Centre for Mathematics and its Applications, School of Mathematical Sciences,
Australian National University, ACT 0200, Australia

Abstract

We present a "practical" algorithm to construct random elements of a finite group. We analyse its theoretical behaviour and prove that asymptotically it produces uniformly distributed tuples of elements. We discuss tests to assess its effectiveness and use these to decide when its results are acceptable for some matrix groups.

1 Introduction

How do we select random elements from a large finite group G? Our answer will depend on the type of description we have for the group. If G is a permutation group, we can construct a base and strong generating set; if G is a finitely-presented soluble group, we can obtain a power-conjugate presentation. In both cases, we can use these special descriptions to obtain random elements.

Our algorithm is designed for the case where G is described by a generating set X, and we have no convenient canonical form for the elements of G. In particular, we consider the case where G is a subgroup of $GL(d, q)$, the group of non-singular $d \times d$ matrices defined over $GF(q)$, for q a prime-power. We seek to develop a "practical" algorithm for matrix groups having degrees up to the hundreds.

Since elements of G can only be constructed as words in X, the problem is to construct words that will define random elements of G. We face a fundamental problem: the cost of a single matrix multiplication is $O(d^3)$. Hence our requirement of practicality dictates that we perform only a small number of matrix multiplications. However, Holt & Rees (1992) demonstrate that words of length up to 20 in the supplied generators are far from random.

Babai (1991) proposed a general solution to the problem. Let n be an upper bound for the order of G. His algorithm finds a set of $O(\log n)$ elements in $O((\log n)^5)$ multiplications. Using this set, a sequence of nearly uniformly distributed random elements can then be obtained in $O(\log n)$ multiplications for each element. Since $\log n \leq d^2 \log q$, the cost to obtain the set can be $O(d^{10}(\log q)^5)$. Babai's primary aim was to provide an algorithm which is guaranteed to produce nearly uniform elements with large probability within $O((\log n)^c)$ number of steps, for some constant c. Beals implemented a version of this algorithm, incorporating various heuristic shortcuts, and it is used as a component in his implementation of the algorithm of Babai, Beals & Rockmore (1993) to decide finiteness of a matrix group over a number field. Babai (personal communication) reports that he and Andras Lukacs have developed another algorithm for abelian groups: from a given set of k generators, this algorithm reaches near uniformity in $O(k \log n)$ steps.

Diaconis & Saloff-Coste (1993 and 1994) consider symmetric random walks on finite groups, and their rate of convergence to the uniform distribution.

In this paper we present an algorithm — the *product replacement* algorithm — that generates "random" elements of a finite group efficiently. We specify the parameters which influence its performance and present the algorithm in sufficient detail to permit a reader to develop an independent implementation. It is a *black box* algorithm, following the terminology of Babai (1991), requiring only the ability to multiply group elements. It generates N-tuples of elements, where N is a positive integer. We prove that asymptotically these N-tuples are uniformly distributed among all N-tuples from G which contain a generating set for G. Our algorithm has some common features with Markov chain Monte Carlo methods.

At this point, it is worth considering the primary motivation for our work: namely, its application to the matrix group "recognition" project. Most matrix group algorithms currently in use rely on first obtaining a permutation representation of the group and then using highly-developed permutation group machinery to carry out structural investigations. In practice, many structural questions cannot be answered for an arbitrary matrix group because a "moderate" degree permutation representation either does not exist or cannot easily be found. Aschbacher (1984) classified the subgroups of $GL(d, q)$ into nine categories. His work has provided the theoretical framework for a project which seeks to develop a "second generation" of matrix group algorithms which use the inherent matrix structure of the group. We can summarise his classification as follows: a matrix group is almost simple modulo scalars, or it preserves some natural linear structure and has a normal subgroup related to this structure. The first step in "recognising" a matrix group is to determine (at least one of) its categories in the Aschbacher classification. If a category of the group can be recognised, we hope to investigate its structure more completely using algorithms designed specifically for that category.

Algorithms have been developed to recognise some of the Aschbacher categories. All assume that elements of a matrix group which satisfy certain properties can be obtained efficiently. For example, the algorithm for imprimitivity testing of Holt, Leedham-Green, O'Brien & Rees (1994) uses orders and the structure of characteristic polynomials of randomly-selected elements to rule out certain possible block sizes. Celler & Leedham-Green (in preparation) use

element orders and minimal polynomial structure to decide whether a group contains a classical group. In such cases, it is desirable to select at least one element for each possible value of the chosen property. In addition, some of the algorithms are Monte Carlo in character. The recognition algorithm for special linear groups of Neumann & Praeger (1992) searches for (nearly) irreducible elements in the supplied group. The algorithms for recognising classical groups of Niemeyer & Praeger (in preparation) search for elements which satisfy the *primitive prime divisor* property. In each case, they have determined the proportions of these types of elements in the classical groups. If none is found after a number of random selections, they conclude with a certain probability that the given group does not contain a classical group. The excellent theoretical behaviour of such algorithms is achievable in practice if we have tools available to generate elements with certain properties in the correct proportion.

Given this motivation, we have two fundamental aims: we seek to generate elements rapidly and to ensure that they are well-distributed according to various predicates or criteria. Hence, we design experiments to measure when the results of our random element generator are acceptable. How do we decide that the elements are sufficiently random? The only black box test of a random selector from a very large and featureless set is that it "never" select the same element twice. In practice, we use properties of fundamental importance to the recognition project — element order distributions; the proportion of cyclic matrices; and the proportion of primitive prime divisor elements — to assess the quality of the output, applying χ^2-tests to decide when our selections are acceptable. While our algorithm does not produce uniformly distributed random elements, we conclude that it achieves our central aims for these important properties.

The structure of the paper is as follows. We first present the algorithm and analyse its theoretical performance. In Section 4 we discuss the use of statistical tests to assess the quality of a random element generator. In Section 5 we explain the specific tests used to assess our algorithm. Finally we report on the performance of an implementation.

2 The algorithm

Let a group G be described by a generating set $X = \{X_1, \ldots, X_k\}$. We choose an integer $N > k$ and *initialise* an array S of length N to consist of the generators of G, where we allow repetitions. The *basic operation* of the algorithm takes a pair of random integers $i \neq j$ in $[1, \ldots, N]$, and replaces $S[i]$ by $S[i]S[j]$ or $S[j]S[i]$. We carry out a *preprocessing step* by executing this basic operation a number of times, K. We now execute the basic operation and return the resulting value of $S[i]$ as the random element.

Note that S at all times contains a generating set for G. This method has the advantage that, after the preprocessing, only one multiplication is required for each random element. In addition, since we replace $S[i]$, we hope that the process of finding new random elements increases the randomness of S. Since the effective cost of the algorithm is $K\,O(d^3)$, we hope to demonstrate by experiment that, for some small value of K (preferably independent of the degree d), the elements obtained are sufficiently random for our purposes.

In practice, we make a random choice of whether to replace $S[i]$ by $S[i]S[j]$ or $S[j]S[i]$, and our later discussion is independent of this. The key choices in this algorithm are the number of basic operations, K, which must be carried out as part of the preprocessing to obtain a reasonable distribution, and the length, N, of S.

One obvious disadvantage of the technique is that the elements returned are not independent of each other. For example, if a sequence of elements is generated, then a consecutive triple of the form a, b, ab will occur in the sequence with probability greater than $1/N^2$.

The algorithm presented here is based on an idea of Leedham-Green and Soicher. Holt & Rees (1992) use a variation of this technique, which can be obtained by a suitable choice of parameters K and N. In their version, the supplied generating set is first enlarged by adding $\max(k, 10)$ new elements to give N generators in all; the new generators are constructed by taking words of length about 30 in the supplied generators. About N^2 basic operations are carried out in preprocessing the array S.

3 An analysis of the algorithm

Recall that S is the array constructed during the initialisation part of the algorithm to store copies of the supplied generating set of G. Its contents are modified by the execution of the algorithm. We study the behaviour of the algorithm and prove that asymptotically the probability distribution of S tends exponentially to the uniform distribution.

Theorem 1 *Let m be the maximal cardinality of a minimal generating set X of G, and let $N \geq 2m$. Let Y be the set of ordered N-tuples of G that generate G. Let S_t be the element of Y obtained by repeating the basic operation t times. For each $v \in Y$ the probability $p_t(v)$ that $S_t = v$ tends to $1/|Y|$ as t tends to infinity.*

Proof. The sequence p_t is a Markov chain. More formally, we have $\mathrm{P}(S_{t+1} = v|S_t) = \mathrm{P}(S_{t+1} = v|S_1, \ldots, S_t)$; that is, S_{t+1} is obtained from S_t without reference to earlier entries in the chain. Note also that the chain is homogeneous; that is, this probability is independent of t. Following standard practice, we shall refer to the possible values of S_t — that is, the elements of Y — as states, and t as time.

The first step in proving that the above process behaves as claimed is to prove that any two states intercommunicate; that is, given states v and w, there is a time t such that one can move from v to w with positive probability in time t.

Assume first that $N = 2m$. Let S_a and S_b be states. It is easy to see (since every element of G has finite order) that, if there is a positive probability of moving from S_a to S_b in time t, then there is a positive probability of moving from S_b to S_a (perhaps requiring time greater than t). Thus it suffices to find a state S_c such that, for t large enough, there is a positive probability of moving from S_a to S_c and from S_b to S_c in time t. Now S_a contains a generating set $x_1, \ldots, x_m$ say, and S_b contains a generating set $y_1, \ldots, y_m$ say. Clearly, we can get from S_a to a sequence consisting of $x_1, \ldots, x_m, y_1, \ldots, y_m$ in some order. Similarly we can get from S_b to a similar sequence. It remains to prove that we can perform any permutation of such a sequence. Since $x_1, \ldots, x_m$ generate G, it is clearly possible to permute any two y_is, and similarly we can permute any two x_is. Suppose now that we wish to interchange x_i and y_j. Since $y_1, \ldots, y_m$ is a generating set for G, we may replace x_i by $x_i y_j^{-1}$. We

may then multiply y_j on the right by $x_i y_j^{-1}$, thus replacing y_j by x_i. Since $x_1, \ldots, x_m$ is a generating set for G, we can now replace $x_i y_j^{-1}$ by y_j. The general case where $N \geq 2m$ should now be clear. A chain in which any two states intercommunicate is said to be irreducible.

We will now prove that the states are aperiodic; that is, for each state v, the greatest common divisor of the integers t such that there is a positive probability of the chain returning from state v to v again in t steps is 1. This is clear, since, with positive probability, we may pass from any state back to itself going through a state in which one component is the identity, and in this case we may add 1 to the length of the chain back to our original state by vacuously multiplying an entry by the identity.

Finally, we prove that the equilibrium distribution for this Markov process is the uniform distribution. Since the chain is homogeneous, irreducible, and aperiodic, it suffices to prove that we have a doubly stochastic process; that is, the matrix whose rows and columns are each labelled by the possible states, and whose (u, v) entry is the probability that the $t+1$ state will be v, given that the t-th state is u, has entries whose row and column sums are all equal to 1. The fact that the row sums are 1 is automatically satisfied for any stochastic process. If we define a new stochastic process in which we choose, with equal probability, any ordered pair (u, v) with $1 \leq u \neq v \leq N$, and replace $S[i]$ by $S[i]S[j]^{-1}$, then this is the reverse of the given stochastic process, and its transition matrix is the transpose of that for the original stochastic process. Since the new matrix has its row sums equal to 1, the original matrix has its column sums equal to 1 and the result follows. □

It is easy to obtain a qualitative result on the speed of convergence. The transition matrix p of the above Markov process is defined by $p_{uv} = p_{uv}(1)$, so $p_{uv}^t = p_{uv}(t)$. Now p has one eigenvalue equal to 1, and the other (complex) eigenvalues have modulus strictly less than 1. This is the Perron-Frobenius theorem, which applies to Markov chains with finitely many states; see Grimmett & Stirzaker (1982, p. 134). It proves the next result.

THEOREM 2 *Let M be $|Y|$. Then for some positive $\epsilon < 1$, for all states u and v, and for t sufficiently large, $|p_{uv}(t) - 1/M| < \epsilon^t$. That is, whatever the initial state, the probability of having any state after time t tends exponentially to the uniform distribution.*

Although in the limit each element of Y is equally likely, this does not imply that the process will yield each element of G with equal probability. The question which arises is to determine the number of elements of Y that contain a specified element g of G in some specified place. What can be proved about the proportion of $(N-1)$-tuples of G that generate G? Recent examples of results in this direction are provided by Kantor & Lubotzky (1990) and Liebeck & Shalev (to appear); they show that the probability that a pair of random elements of a simple group generates the group tends to one as the order of the group tends to infinity. If the proportion of $(N-1)$-tuples which generate G is very close to one, then all elements of G will be almost equally likely to occur, with a slight bias, for example, away from the identity. Clearly the number of tuples in Y containing g in some specified place depends only on the conjugacy class of g (in fact, on its conjugacy class in the holomorph of G); so it is sufficient to estimate the frequency with which representatives of the various conjugacy classes of G arise.

Our final objective is to make this theorem quantitative. In practice, our main concern is to bound the number of basic operations needed to give a reasonably uniform distribution of elements. While the prospect of proving realistic bounds seems small, in Section 6 we provide partial answers to this question.

Clearly, the worst case would arise if the set of states was partitioned into two subsets, with very few processes taking one from the first to the second. If the initial state is in one of these subsets, it may take a long time before there is a reasonable probability of the current state being in the other subset. It seems unlikely that this can occur.

4 Testing randomness

In Section 3 we prove that the algorithm generates N-tuples which asymptotically exhibit uniform distribution. Our concern is now to decide whether this good behaviour is reflected in the distribution of the generated elements.

Our task is to test the hypothesis that an algorithm generates uniformly distributed group elements. An easy test is the following: choose an integer n which is very large compared to the order of G; generate n elements of G; count how often each element is encountered. Since we wish to investigate

groups of both large order and large degree, this approach has limited value.

Niederreiter (1992, Chapter 7) suggests that a good method of testing pseudo-random number generators is to apply statistical tests to them. We follow his advice and apply three χ^2-tests to the results of our algorithm. Holt & Rees (1992) also used this statistical test to examine the performance of their method.

We first choose a set, $\{P_1, \ldots, P_b\}$, of properties where all elements of a group satisfy one of these and no element satisfies more than one. That is, we partition the group into b classes. Let G be a group where we know the number p_i of elements that have property P_i for $1 \leq i \leq b$. If we have n uniformly distributed random elements, then the expected number, e_i, of elements that have property P_i is $np_i/|G|$. We use the random element generator to obtain n elements of G and determine the number, n_i, of these that have property P_i. We can now compute a χ^2 value for our data. Recall that χ^2 is defined by

$$\chi^2 = \sum_i \frac{(n_i - e_i)^2}{e_i}.$$

Associated with the χ^2-test are two parameters: the *number of degrees of freedom* and a *χ^2-probability*. The number of degrees of freedom is the number of independent outcomes. Since the sum of the n_i is n, there are at most $b-1$ degrees of freedom. The χ^2-probability, α, is chosen to be small, usually in the range from 0.01 to 0.1. When both parameters are chosen, we can determine the *critical value*, x, such that the probability that a χ^2 random variable, having b degrees of freedom, exceeds x is α. In particular, we can carry out a χ^2-test on a number of independent data sets. If the results of the tests exceed the critical value more than the number of times determined by our chosen χ^2-probability, we reject our hypothesis. This is the "Neyman Pearson" method of hypothesis testing. Tables of critical values for various degrees of freedom and χ^2-probability are available in the literature.

However, one test of this type is not sufficient to decide that the elements generated are evenly distributed in the group. For example, assume we choose just one property P and we test a group having four elements, two of which have property P. Let g be one of these and let h be an element that does not have property P. An algorithm that returns either g or h each with probability 0.5 would pass the χ^2-test. In an attempt to address the problem

that a chosen test may have little power against certain alternative hypotheses, we test three sets of properties on independent data. Assume, without loss of generality, that we fix α as the χ^2-probability for each of these tests. If the hypothesis that the algorithm produces uniformly distributed elements is true, then the probability that all three χ^2-tests have values greater than their respective critical values is α^3 and the probability that one test exceeds the critical value is $1-(1-\alpha)^3$. For example, if $\alpha = 0.05$ this probability is 0.14.

5 The properties

We now describe the group-theoretic properties used to test our algorithm. Recall that a set of properties is used to partition the group. We choose properties which play a key role in the algorithms developed as part of the matrix group recognition project. Our motivation is two-fold: we want to ensure that our algorithm performs well for these properties and theoretical estimates for the expected results are available.

The first property we test is whether the generated elements have particular orders. In practice we know — from direct computation or tables of data such as the Atlas of Conway *et al.* (1985) — the frequency distribution of orders for various groups.

An element of $GL(d,q)$ is *cyclic* if its characteristic and minimal polynomials are identical. Neumann & Praeger (in preparation) estimate the proportion of cyclic matrices in the general linear group and classical groups. They expect that these elements will play an important role in deciding whether a matrix group preserves a bilinear or sesquilinear form. We test whether the elements generated are cyclic.

For integers $b, e > 1$, a *primitive prime divisor* of $b^e - 1$ is a prime dividing $b^e - 1$ but not dividing $b^i - 1$ for any $1 \leq i < e$. An element of $GL(d,q)$ whose order is divisible by a primitive prime divisor of $q^e - 1$, where $d/2 < e \leq d$, is a *primitive prime divisor element* (ppd-element). Penttila, Praeger & Saxl (in preparation) classify subgroups of the general linear group which contain ppd-elements. These elements play a fundamental role in the work of Niemeyer & Praeger (in preparation) to recognise whether a subgroup of $GL(d,q)$ contains a classical group and so they computed the proportion of ppd-elements in classical groups. We test whether the elements generated are ppd-elements.

We now present the theoretical estimates for the proportion of elements in various groups that satisfy the last two properties. We first need to introduce some notation from Kleidman & Liebeck (1990). Let V be the d-dimensional vector space over $GF(q)$. Let κ denote a non-degenerate form on V. Then Δ denotes the subgroup of $GL(d,q)$ that consists of all matrices which leave κ invariant up to scalar multiplication and I denotes the subgroup of Δ that leaves κ invariant. Let S denote the intersection of I and $SL(d,q)$. Then Ω is the subgroup S, except if κ is a non-degenerate quadratic form, in which case Ω has index 2 in S. Note that if $\kappa \equiv 0$ then $\Delta = GL(d,q)$ and $\Omega = SL(d,q)$; if κ is a non-degenerate symplectic form then $\Omega = Sp(d,q)$; if κ is a non-degenerate unitary form then $\Omega = SU(d,q)$; if κ is a non-degenerate quadratic form then $\Omega = \Omega^{\epsilon}(d,q)$. The orthogonal case has three subcases according to the type of standard basis for V (see Proposition 2.5.3 in Kleidman & Liebeck); these are indicated by ϵ being one of $\circ$, $+$ or $-$.

5.1 Cyclic matrices

Neumann & Praeger (in preparation) estimate the probability, ν_G, that a random matrix in G, where $\Omega \leq G \leq \Delta$, is not cyclic. We summarise relevant parts of their results:

(i) Special linear group:

$$\nu_G < \frac{1}{q(q^2-1)} + \frac{1}{q^6}$$

(ii) Symplectic groups:

$$\nu_G < \frac{3}{q(q^2-1)} + \frac{1}{q^3(q-1)}$$

(iii) Orthogonal groups where $d \geq 3$, $p \neq 2$:

$$\nu_G < \frac{2q^2+q+1}{q(q^2-1)} + \frac{3}{2q(q-1)} + \frac{1}{q^2(q-1)^2}$$

(iv) Orthogonal groups with $p = 2$:

$$\nu_G < \frac{q}{(q^2-1)} + \frac{1}{q(q-1)} + \frac{3}{2q(q^2-1)} + \frac{1}{2q^2(q-1)^2}$$

5.2 Primitive prime divisors

Let $Pr(G)$ denote the probability of obtaining a ppd-element by a single random selection from G. We summarise relevant parts of Niemeyer & Praeger's results. We assume that $\Omega \leq G \leq \Delta$.

(a) In the symplectic case d is even, $d \geq 4$, and

$$\sum_{a \leq f \leq d/2} \frac{1}{2f+1} \leq Pr(G) < \sum_{a \leq f \leq d/2} \frac{1}{2f},$$

where $a = (d+4)/4$ if $d \equiv 0 \pmod 4$ and $a = (d+2)/4$ if $d \equiv 2 \pmod 4$.

(b) In the unitary case

$$\sum_{a \leq f \leq b} \frac{1}{2f+2} \leq Pr(G) < \sum_{a \leq f \leq b} \frac{1}{2f+1},$$

where a and b are integers with $(d-1)/4 \leq a \leq (d+2)/4$ and $(d-2)/2 \leq b \leq (d-1)/2$.

(c) In the orthogonal case with q odd:

(i) For d odd, $d \geq 7$,

$$\sum_{a \leq f \leq (d-1)/2} \frac{1}{2f+1} \leq Pr(G) < \sum_{a \leq f \leq (d-2)/2} \frac{1}{2f} + \frac{2}{d-1},$$

where a is an integer with $(d+1)/4 \leq a \leq (d+3)/4$.

(ii) For d even, $d \geq 6$, set $\delta = 0$ in the $+$ case and $\delta = 1$ in the $-$ case. We obtain

$$\sum_{a \leq f \leq (d-2)/2} \frac{1}{2f+1} + \frac{\delta}{d+1} < Pr(G) < \sum_{a \leq f \leq (d-2)/2} \frac{1}{2f} + \frac{2\delta}{d},$$

where $a = (d+4)/4$ if $d \equiv 0 \bmod 4$ and $a = (d+2)/4$ if $d \equiv 2 \bmod 4$.

6 Investigating the algorithm

We developed an implementation of the algorithm in GAP 3.4 (see Schönert *et al.*, 1994). Our algorithm has two parameters: K is the number of basic operations carried out during preprocessing and N is the size of the array S.

We also select the following: the property used to test the output; the number, r, of independent executions of the algorithm to perform; the number, n, of selections to perform during each execution.

Assume that the number of possible outcomes of the tests is b. The implementation constructs an $n \times b$ array, R. For each of the r executions, the implementation records the outcome of selection i in the ith row of R. Hence at the end of r executions, the ith row of R records the distribution of r elements, all obtained after $i + K$ basic operations. We now apply a χ^2-test to each of the n distributions of r elements stored in R.

Below, we present our results for a range of examples. We have chosen commonly available generating sets for these groups — primarily those supplied by GAP or Magma (Bosma & Cannon, 1994). Let k be the size of the supplied generating set X. We choose N to be the maximum of $2k + 1$ and 10, and use a χ^2-probability of 0.05.

We choose K to be zero for our tests — that is, we only initialise S and do not perform any basic operations on its contents. This allows us to decide most easily when the number of basic operations is sufficient to provide a "reasonable" distribution. What constitutes an acceptable distribution? Since we use a χ^2-probability of 0.05, we record the smallest number, K, of basic operations where at most 5% of the remaining $n - K + 1$ χ^2-tests exceeds the critical value.

Table 1: Order test for a sample of groups

Group	Order	K
J_2	604800	51
$PSp(6, 2)$	1451520	48
$U_5(2)$	13685760	56
A_{11}	19958400	71
HS	44352000	49
M_{24}	244823040	57
S_{12}	479001600	53

In Table 1, we report on our testing of order distributions. For each group, we list its (Atlas or well-known) name and its order. We also list the number,

K, of basic operations performed before the distribution settles. Since the algorithm is black box, in some cases we used permutation (rather than matrix) representations for the groups. The values of n and r are 150 and 50 000, respectively. Since we know the order distributions for our groups, it is easy to run a χ^2-test on the results.

When we carry out our ppd-element and cyclic matrix tests, we have to work a little harder to apply a χ^2-test to the results. Since we know only a range for the expected outcome, we must first estimate its value. We then hypothesise that this estimate is the expected outcome and use our χ^2-test to test this hypothesis. We obtain an estimate of the expected outcome by generating a large number of elements — about 50 000 in each group — and determining the proportion of these which satisfy each property. If the computed value is within the theoretical range presented in Section 5, we use it as the expected value for the χ^2-test.

Table 2: Cyclic matrix and ppd-element tests for a sample of groups

Group	K_c	K_p
$Sp(10, 16)$	18	24
$Sp(30, 7)$	77	19
$SU(20, 25)$	–	36
$SU(30, 7)$	–	60
$O^+(10, 25)$	33	18
$O^-(10, 25)$	33	31
$O^-(20, 7)$	52	31
$O^+(30, 8)$	29	99

In Table 2, for a range of classical groups we report the numbers, K_c and K_p, of basic operations performed for the cyclic matrix and ppd-element tests before the distribution settles. The values of n and r are 150 and 2 000, respectively.

What can we say about the behaviour of K if $|X|$ is fixed and $|G| \to \infty$? For example, consider the behaviour of K for the family of general linear groups, $GL(d, q)$, as $d \to \infty$. Since the order of $GL(d, q)$ is about q^{d^2}, and each basic operation gives rise to one of at most $2N(N-1)$ elements

of G, we need at least $d^2 \log q / \log(2N(N-1))$ basic operations to cover the whole group. Hence, theoretically, we expect that K grows as $d^2 \log q$. We investigated the performance of K for a small sample of general linear groups having degree up to 100 and for a fixed field. We chose two sets of properties for our investigation:

1. The degree of the largest irreducible factor of the characteristic polynomial of an element.

2. The number of irreducible factors of the characteristic polynomial of an element.

The first of these properties (for the minimal polynomial) was also used by Holt & Rees (1992) to assess the quality of their results. Here, as in the case of ppd-elements and cyclic matrices, we first estimated the expected outcome by considering a large sample. In Table 3, for a range of general linear groups we report the numbers, K_d and K_f, of basic operations performed for these two tests before the distribution settles. The values of n and r are 150 and 10 000, respectively.

Table 3: Characteristic polynomial tests for a sample of general linear groups

Group	K_d	K_f
$GL(50, 7)$	45	42
$GL(60, 7)$	49	56
$GL(70, 7)$	58	67
$GL(80, 7)$	59	69
$GL(90, 7)$	65	76
$GL(100, 7)$	83	86

It appears that, within the *range of our experiment*, the value of K is bounded or at most grows slowly. The fact that a much smaller bound for K appears to hold than that expected on purely theoretical grounds suggests that the partitions chosen in our experiments are reasonably evenly distributed among words of different length. However, we would need to investigate matrix groups having degrees up to the thousands before being able to reach more decisive conclusions and currently such investigations are not practical.

What influence does the size N of the array, S, have on performance? In practice, we cannot ensure that N is at least twice the maximal cardinality of a minimal generating set for G. Hence, Y consists of N-tuples containing only generating sets in the same Tietze class as the original generating set, X. Let X have cardinality k. We found that the performance varied little provided that N was a "small" multiple of k, usually no more than five. If N is larger than this value, then the number of basic operations required for preprocessing significantly increases — presumably because the array contains too many repetitions. We observed best overall performance when we chose the array size to be the maximum of $2k + 1$ and 10.

We have considered briefly the influence of the initial generating set on the performance on the algorithm. It is not possible to carry out a systematic study since there is no agreed notion of either a "bad" or "random" generating set. In practice, common generating sets for linear groups are far from random. Our investigation suggests that the impact is the natural one: namely, not all generating sets require the same amount of preprocessing. For example, we chose two generating sets for S_{10}: one of cardinality 9 composed entirely of transpositions, and another of cardinality 2 containing a transposition and an element of order 10. The number of basic operations required for these generating sets was 109 and 63 respectively. The equivalent investigation for S_8 gave values of 52 and 43.

We conclude that the product replacement algorithm, with parameter settings $N = \max(10, 2k+1)$ and $K \geq 60$, appears to give adequate performance for this collection of properties.

Versions of the algorithm are available as part of the standard distributions of GAP and MAGMA. We have also written a library of procedures to test the performance of random element generators, which is available on request. Apart from its use for additional testing, it may be useful in developing Monte Carlo algorithms. If an algorithm returns an answer based on a certain theoretical probability, then our programs can be used to check whether the sample chosen in a particular experiment meets the theoretical requirements.

Acknowledgements

We thank the following colleagues for very helpful discussions and feedback: László Babai, Adrian Baddeley, Gene Cooperman, Larry Finkelstein, Derek F. Holt, Cheryl E. Praeger, and Sarah Rees. Leedham-Green, Murray, and Niemeyer thank the School of Mathematical Sciences at the Australian National University for its hospitality while part of this work was carried out. The work of Niemeyer on this project was supported by ARC Grant A69230241. O'Brien thanks both the University of Western Australia and Northeastern University for their hospitality while this work was being completed. The computations reported in Table 3 of the paper were carried out using GAP-MPI, a parallel version of GAP, currently under development by Cooperman at Northeastern University.

References

References

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

László Babai (1991), "Local expansion of vertex-transitive graphs and random generation in finite groups", *Theory of Computing*, (Los Angeles, 1991), pp. 164–174. Association for Computing Machinery, New York.

László Babai, Robert Beals and Daniel Rockmore (1993), "Deciding finiteness of matrix groups in deterministic polynomial time", *Proc.* 1993 *International Symposium on Symbolic and Algebraic Computation*, pp. 117-126. ACM Press, New York.

Wieb Bosma and John Cannon (1993), *Handbook of* Magma *functions*. Department of Pure Mathematics, Sydney University.

Frank Celler and C.R. Leedham-Green (in preparation), "Non-constructive classical group recognition".

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

Persi Diaconis and Laurent Saloff-Coste (1993), "Comparison techniques for random walk on finite groups", *Ann. Probab.*, **21**, 2131–2156.

P. Diaconis and L. Saloff-Coste (1994), "Moderate growth and random walk on finite groups", *Geom. Funct. Anal.*, **4**, 1–36.

Geoffrey Grimmett and David Stirzaker (1982), *Probability and Random Processes*. Oxford University Press, London.

Derek F. Holt, Charles R. Leedham-Green, E.A. O'Brien and Sarah Rees (1994), "Primitivity testing for matrix groups", preprint.

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

W.M. Kantor and A. Lubotzky (1990), "The probability of generating a finite classical group", *Geom. Dedicata*, **36**, 67–87.

Peter Kleidman and Martin Liebeck (1990), *The Subgroup Structure of the Finite Classical Groups*, London Math. Soc. Lecture Note Ser., **129**. Cambridge University Press.

Martin W. Liebeck and Aner Shalev (to appear), "The probability of generating a finite simple group", *Geom. Dedicata*.

Peter M. Neumann and Cheryl E. Praeger (1992), "A recognition algorithm for special linear groups", *Proc. London Math. Soc.* (3), **65**, 555–603.

Peter M. Neumann and Cheryl E. Praeger (in preparation), "Cyclic matrices over finite fields".

Harald Niederreiter (1992), *Random Number Generation and Quasi-Monte Carlo Methods*, CBMS-NSF Regional Conference Series in Applied Mathematics, **63**. SIAM, Philadelphia.

Alice C. Niemeyer and Cheryl E. Praeger (in preparation), "Recognising classical groups".

Tim Penttila, Cheryl E. Praeger and Jan Saxl (in preparation), "Linear groups with orders divisible by certain large primes".

Martin Schönert *et al.* (1994), GAP – *Groups, Algorithms and Programming.* Lehrstuhl D für Mathematik, RWTH, Aachen.