

Product Replacement in the Monster

Petra E. Holmes
School of Mathematics and Statistics,
University of Birmingham,
Edgbaston, Birmingham B15 2TT,
United Kingdom.

Stephen A. Linton
School of Computer Science,
University of St. Andrews,
North Haugh, St. Andrews, Fife, KY16 9SS
United Kingdom.

Scott H. Murray
Faculteit Wiskunde en Informatica,
Technische Universiteit Eindhoven,
Postbus 513, 5600 MB Eindhoven,
Nederland

October 17, 2003

Abstract

We show that the product replacement algorithm can be used to produce random elements of the Monster group. These random elements are shown to have the same distribution of element orders as uniformly distributed random elements after a small number of steps.

1 Introduction

Computing in finite groups often requires a supply of random elements. There are several known methods for producing them. The best known practical method is the product replacement algorithm which is given in Section 2.

The Monster is the largest of the 26 sporadic simple groups. It has order

808 017 424 794 512 875 886 459 904 961 710 757 005 754 368 000 000 000

and its minimal faithful permutation and matrix representation degrees are respectively 97 239 461 142 009 186 000 and 196 882. This makes it far harder to work with than the 25 smaller sporadics.

In Section 3, we describe techniques which allow limited computation in the Monster. In Section 4 we describe our experiments to assess the effectiveness of combining these techniques with a suitable version of the product replacement algorithm, and in Section 5 we present our results and conclusions.

2 The product replacement algorithm

Let G be a finite group generated by the set X . The product replacement algorithm uses an array $s = (s_1, \dots, s_m)$ of elements of G satisfying the property that $\langle s_1, \dots, s_m \rangle = G$. We require that m be larger than the size of X . Initially we take the entries in s to be the elements of X , with repetitions to fill out the array. Then at each stage of the algorithm, we choose distinct random integers i and j between 1 and m ; and then replace s_i by the product $s_i s_j$. We then return the new value of s_i as our random element. It is known from [2] that the random elements returned converge in the long term to a fixed distribution. While this distribution is often close to uniform, [7] has shown that in certain cases it can be very far from uniform.

A more recent variant of this algorithm [5] does converge to the uniform distribution. In this variant we have an extra group element s_0 , called the *accumulator*. Then at each step we do product replacement as above, but in addition we choose a third random integer k and multiply s_0 by s_k . Then s_0 is returned as our random element.

We have run 100 tests of the product replacement algorithm (with and without accumulator) with different random seeds. We assess the randomness of the elements produced using the χ^2 statistic at the 0.9 probability level, applied to a test value derived from the orders of the random elements. We consider that the algorithm has converged at step t if, for at least nine out of the subsequent ten steps, the χ^2 value is below the 0.9 level, as would be expected from genuinely random elements. These methods for testing product replacement are based on [1, 2].

3 Computing in the Monster

Computing in the Monster is very different to computing in smaller groups. In most of the groups that we work in, generating elements can be stored as permutations or matrices. In the Monster we must take a different approach, as seen in the three constructions of [6], [4] and [10]. The construction of [6], which uses 3-local subgroups and linear algebra in characteristic 2, gives us the fastest way of computing in \mathbb{M} and so we choose it for our computations.

All three of the constructions use three generators, two of which generate a local subgroup. In the construction of [6], two of the generators: C and D , generate a subgroup isomorphic to $3^{1+12} \cdot 2\text{Suz}:2$, the normaliser of an element in class $3B$. The third generator, T , normalizes a subgroup of $\langle C, D \rangle$ isomorphic to $3^{2+5+10} : (\text{M}_{11} \times 2^2)$, the centraliser of two commuting elements of class $3B$. The

element T extends this group to $3^{2+5+10}:(M_{11} \times D_8)$. The dimension 196882 module for \mathbb{M} over $GF(2)$ restricts to $3^{1+12} \cdot 2\text{Suz}:2$ with shape

$$142 \oplus 32760 \oplus (3^6 \otimes 90),$$

where all dimensions are over $GF(4)$ except 142. So C and D are represented as files each containing four matrices, one for each piece of the representation. Similarly, the module structure for $3^{2+5+10}:(M_{11} \times D_8)$ allows T to be stored as a collection of small matrices.

There are two main programs: one which calculates the image of any vector in the 196882-dimensional space under any of the generators, and one which can multiply together the elements in the local subgroup and return the product in the same format as the generators. In this paper we use the vector-image program to calculate orders of words in the generators.

It takes approximately 60 ms for each occurrence of T in a word to multiply a vector by a word using a Pentium II/450MHz processor with 384 MB of RAM. This operation is about 100 times faster than when using the construction of [4], although that construction is the more frequently used as it gives easy access to 2-local subgroups and comes equipped with a method for shortening words [3].

4 Product Replacement in the Monster

We performed 100 independent incarnations of the product replacement algorithm, both with and without an accumulator, running them for 20 steps with an accumulator and 25 without.

To assess the uniformity of the distribution of random elements that we obtained using the χ^2 test, we need some easily calculated property of these elements which will take a reasonably small number of values and whose distribution in the whole Monster is known. The χ^2 test can then be used to compare the distribution of the values which appear in our sample with the true distribution of the value. In the Monster, the only suitable property which it appears feasible to calculate is element order, which has also been used in studying product replacement in other groups [2]. To meet the requirements of the χ^2 test, that each outcome have expected frequency at least 1 for the sample size being used, we group the orders $\{1, 2, \dots, 23, 25, 28, 33, 34, 44, 45, 55, 105, 110\}$ together to form a single test value.

In fact we do not strictly use element order, but instead use the order of the action of the elements on a fixed test vector. It is highly likely, but not proven, that this vector lies in a faithful orbit of \mathbb{M} . Otherwise there will be a very small chance of obtaining a divisor of the correct order. Since this would make the orders look less, rather than more random, it does not invalidate our conclusions.

The product replacement algorithm was actually performed in the free group on three generators using an array of length 4, resulting, for each random seed in two sequences of words (one obtained using the accumulator, and the other

Number of Steps	χ^2 with accumulator	χ^2 no accumulator
1	670.051	419.923
2	390.119	367.669
3	249.562	300.125
4	163.061	252.426
5	142.792	247.926
6	62.536	274.574
7	65.525	223.413
8	36.882	94.410
9	50.313	131.215
10	37.671	56.936
11	37.680	76.733
12	35.448	52.527
13	57.899	35.812
14	48.167	33.932
15	37.387	38.222
16	34.160	44.395
17	37.068	43.748
18	47.394	49.358
19	30.610	40.715
20	40.744	38.068
21		45.162
22		36.205
23		38.297
24		46.629
25		43.727

Table 1: χ^2 values. The entries in bold show where convergence has occurred.

not). This part of the calculation and the computation of χ^2 values was done in GAP [8].

A C program, using subroutines derived from [6], was then used to compute the orders of these words evaluated at the three generators described in section 3.

5 Conclusions

Table 1 gives the χ^2 values for each step. It can be seen that product replacement with accumulator converges after 8 steps, and for product replacement we need 12.

One use of random element generation in any group is to obtain elements of specific conjugacy classes or orders, for use in subsequent calculations, such as finding standard generators [9]. To assess the suitability of product replacement in the Monster for this purpose, we tabulate in table 2 the number of steps of product replacement with accumulator that we have to perform before we see elements of multiples of each possible order amongst our hundred runs. We are

Number of steps	Element orders
1	1, 2, 3, 4, 5, 6, 7, 8, 10, 12, 14, 15, 20, 23, 28, 30, 56, 60
2	9, 13, 31, 39, 41, 45, 46
3	11, 17, 19, 21, 24, 34, 35, 42, 55, 57, 68, 70, 84, 119
4	16, 22, 25, 26, 32, 33, 40, 47, 50, 52, 66, 78, 95, 104
5	18, 27, 36, 48, 51, 54, 62, 69, 92
6	29, 59, 71, 87, 93, 94
7	38, 44
8	88
9	105, 110

Table 2: Element orders occurring in the output of product replacement

only interested in multiples of element orders because we can get the element of the required order by powering up. It is clear from the table that short words are not suitable for generating random elements as there are some orders of elements which we were unable to find without using many steps.

To assess the cost of computing with random elements produced in this way, we graph in figure 1 the length of the words generated against the number of steps of product replacement. Applying a word of length n to a single vector takes about $60n$ ms on a Pentium II/450 processor.

We conclude that, while it is not practical to use a single product replacement calculation to produce a series of random elements of the Monster as is done in other groups, since the words involved would rapidly grow too long, it would appear that, at least as far as element orders are concerned, starting a fresh product replacement algorithm and running it for between 8 and 15 steps (depending on the quality of randomness required) is both a feasible and an effective way to generate pseudo-random elements of the Monster.

Note that it is easy to see that the elements obtained in this way must be far from uniformly distributed on \mathbb{M} . By counting the possible random choices, we see that n step product replacement with (without) accumulator can produce at most $(4*3*4)^n$ ($(4*3)^n$) distinct outcomes, and this reaches the order of the Monster only for $n \geq 32$ ($n \geq 49$). What our experiments show is that product replacement approaches the same distribution of element orders as the uniform distribution of elements much sooner than this.

6 Acknowledgements

The first author is supported by EPSRC grant GR/R95265/01. The computations described in this paper were performed on equipment provided by EPSRC grant GR/M32351/01.

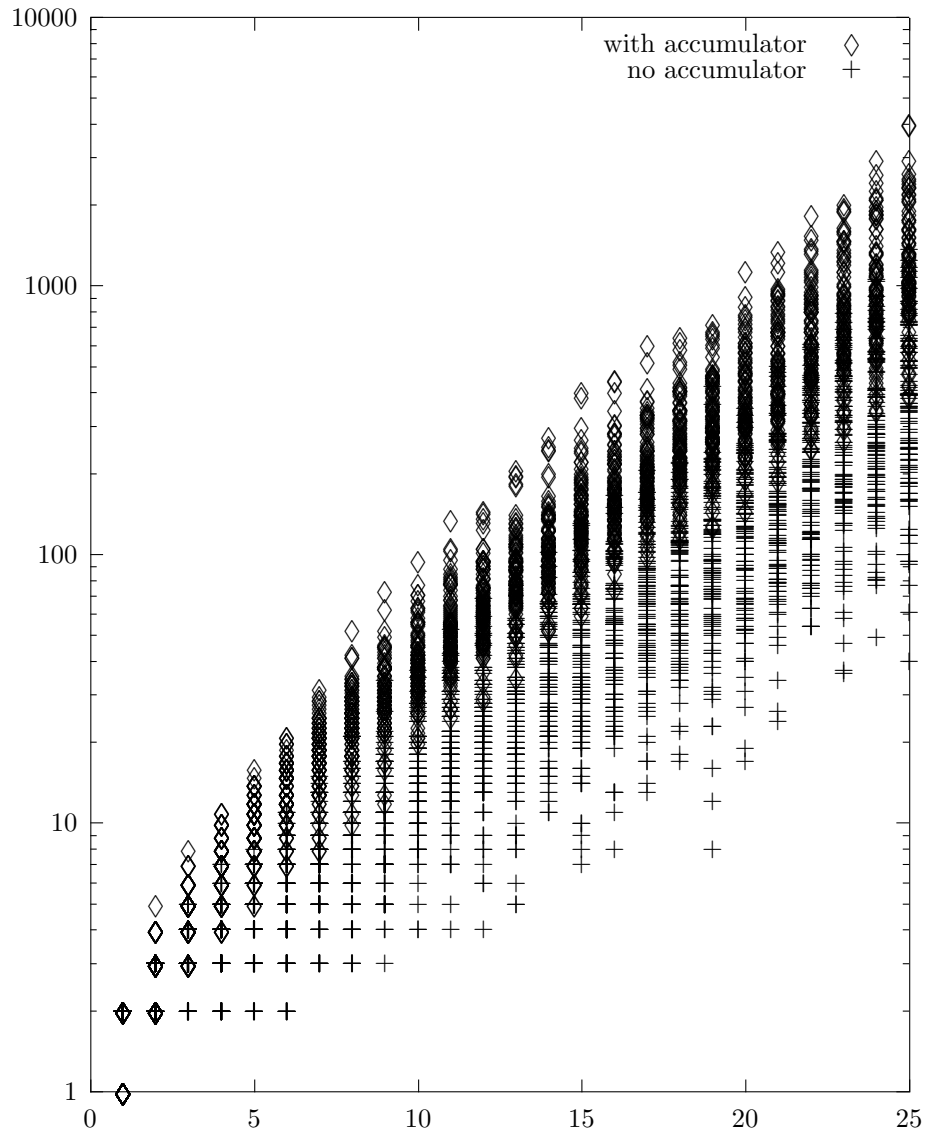


Figure 1: Word Lengths for Product Replacement in the Free Group

References

- [1] László Babai, Walter Kim, Scott H. Murray, and Rebecca Vessenes. Quality of random elements in finite groups. Preprint.
- [2] Frank Celler, Charles R. Leedham-Green, Scott H. Murray, Alice C. Niemeyer, and E.A. O'Brien. Generating random elements of a finite group. *Comm. Algebra*, 23:4931–4948, 1995.
- [3] P.E. Holmes. Computing in the Monster. *PhD Thesis*, Birmingham 2002.
- [4] P.E. Holmes and R.A. Wilson. A new computer construction of the Monster using 2-local subgroups. *J. LMS*, to appear.
- [5] C.R. Leedham-Green and Scott H. Murray, Variants of product replacement, in *Statistical Methods in Group Theory*, Ed. by R. Gilman, A. Myasnikov, V. Shpilrain., Contemporary Mathematics, to appear.
- [6] S.A. Linton, R.A. Parker, P.G. Walsh, and R.A. Wilson. Computer construction of the monster. *J. Group Theory*, 1:307–337, 1998.
- [7] Igor Pak. What do we know about the product replacement algorithm? In *Groups and computation, III (Columbus, OH, 1999)*, pages 301–347. de Gruyter, Berlin, 2001.
- [8] The GAP Group, *GAP – Groups, Algorithms, and Programming*, Version 4.3; 2002, (<http://www.gap-system.org>).
- [9] R.A. Wilson. Standard generators for sporadic simple groups. *J. Algebra*, 184:505–515, 1996.
- [10] R.A. Wilson. A construction of the Monster group over $\text{GF}(7)$, and an application, *Preprint 2000/22*, Birmingham, 2000.