

Calculating the Order of an Invertible Matrix

Frank Celler and C. R. Leedham-Green

ABSTRACT. In the first part of this note we present an algorithm for computing the order of an invertible matrix over a finite field and analyse its complexity. In the second part we compare this algorithm to the so-called spinning algorithm and give variations of the main algorithm to find the projective order and the p' -part, and to decide whether a given prime occurs in the order.

1. Introduction

The object of this note is to observe that it is possible to calculate the order of an element A of $G = GL(d, q)$ on average using $O(d^3 \log q)$ field operations, assuming that $q^i - 1$ has been factorised for $i \leq d$. If this condition is not satisfied, a partial factorisation of the $q^i - 1$ into relatively prime factors p_j can easily be produced, and the integer m is returned, where m is the least product of powers of the p_j that is a multiple of the order of A . This is sufficient for most purposes; for example, it is sufficient for determining whether or not the order of A divides a given integer n . Any ambiguity would arise from the g. c. d. of n and some p_j being a proper factor of p_j . This is readily checked, and if this unlikely event occurs, a factorisation of p_j has been found. Recalculating the order of A with p_j replaced by its newly found factors then resolves the ambiguity.

2. Order Algorithm

The algorithm is based on three simple ideas. The first is as follows.

A BOUNDED ORDER ALGORITHM. *Given an element g of some group, if the order of g is known to divide $n = \prod_i^t p_i^{\alpha_i}$, where the primes p_i are distinct, the order of g can be calculated in at most $2(1 + \lceil \log_2 t \rceil) \lceil \log_2 n \rceil$ multiplications of group elements.*

PROOF. If $t = 1$, just calculate $g^{p_1^j}$ for $j = 1, 2, \dots, \alpha_1 - 1$ or until the identity is obtained. Observe that g^m can always be calculated in at most $2 \lceil \log_2 m \rceil$ multiplications by the standard recursive algorithm: $g^m = g^{m-1} * g$ if m is odd, and $g^m = (g^{(m/2)})^2$ if $m > 0$ is even, see [5] page 441-443. If $t > 1$, then express n as uv , where u and v are coprime, and have almost the same number of distinct

1991 *Mathematics Subject Classification.* Primary 20G40; Secondary 20-04.

prime factors, i. e. differing by at most 1. Now g^u has order k say dividing v , and g^k has order l say dividing u , and the order of g is kl . So the algorithm proceeds by recursion on t . \square

In practice $\log_2 t$ is unlikely to exceed 4 or 5. It is clear that small improvements can be made in the algorithm. For example, if one prime p_i occurs with multiplicity one in n then the algorithm can be organised in such a way that it never has to raise an element to a power involving p_i if p_i doesn't occur in the order and only once if it does. This is extremely helpful if p_i is very large.

REMARK. Note that this algorithm can also be applied if the p_i are relatively prime rather than being prime. In this case, the value m returned as the order is only a multiple of the correct value, but with the property that, for all i , $g^{m/p_i} \neq 1$.

LAS VEGAS TIME. An algorithm has Las Vegas complexity $O(f(n))$ if it will, for any input and any $c > 0$, complete in time at most $ckf(n)$ with probability at least $1 - 1/2^c$, where k is a constant independent of the input and c .

The second idea is as follows. To obtain a multiple of the order of A , first calculate its minimal polynomial.

LEMMA. *The minimal polynomial of $A \in M_d(F_q)$ can be computed using Las Vegas $O(d^3 \max(1, \log_q d))$ finite field operations. The worst case requires $O(d^4)$ finite field operations.*

PROOF. Take any non-zero vector v in the natural module. Let the $F_q A$ -submodule W of V generated by v have dimension k . In order to compute a basis for W form the vectors v, vA, vA^2, \dots reducing them to echelon form, until a linear relation is found. This linear relation describes the minimal polynomial of A restricted to W . Computing the required vA^i needs $2d^2k$ finite field operations, reducing them to echelon form $O(dk^2)$ finite field operations, and keeping track of the echelonised vectors as linear combinations of the original vectors, which is needed to read off the final answer, requires $O(k^3)$ finite field operations. Altogether one needs $O(d^2k + dk^2)$ finite field operations to compute the minimal polynomial of A restricted to W . If W is not the whole of V then one repeats the above calculation starting with a vector $v \notin W$; iterating, a set of polynomials f_i and subspaces W_i are produced, where f_i is the minimal polynomial of A restricted to W_i , until $V = \sum W_i$. Then the minimal polynomial of A is the l. c. m. of the f_i .

The worst situation is when $V = \bigoplus_{i=1}^t W_i$, where each W_i is a reducible uniserial FA -module and t is as large as possible. In this case it is easy to see that the expected time is increased by a factor $O(\log_q d)$. However, in any case after trying d linear independent vector the corresponding W_i span V ; therefore the worst case requires at most $O(d^4)$ finite field operations. \square

REMARK. If $A \in GL(d, q)$ and one is in the bad situation described above, the order of A will be relatively small compared to the order of a general element of $GL(d, q)$. So the bounded order algorithm will almost always run faster in this case than in the general case.

In order to use the bounded order algorithm, one needs to find some n divisible by the order of A . This can be done as follows. Find the degrees d_1, \dots, d_k of the distinct irreducible factors of the minimal polynomial $f(x)$ of A , and the

highest multiplicity α with which such a factor occurs. Then the order of A divides $n = \text{lcm}(q^{d_1} - 1, q^{d_2} - 1, \dots, q^{d_k} - 1) \times p^\beta$, where p is the characteristic of F_q , and $\beta = \lceil \log_p \alpha \rceil$. To see this, reduce A to Jordan normal form over the algebraic closure of F_q . Then each eigenvalue will lie in an extension field of F_q of dimension d_i for some d_i as above, and hence be of multiplicative order dividing $q^{d_i} - 1$. If the block has size $\gamma_i > 1$ then the p -part of the order of the block is p^δ , where $\delta = \lceil \log_p \gamma_i \rceil$, as may be seen by direct calculation. It follows that the above upper bound for the p -part of the order of A is exact, and no powering is needed to obtain the p -part of the order. Finding α involves differentiating $f(x)$ and calculating g. c. d.'s, and is slightly more complicated than in the characteristic zero case, see [5] page 421. It is easy to find the d_i by calculating the g. c. d. of $f(x)$ with $x^{q^j} - x$, for $j = 1, 2, \dots$, where one should divide $f(x)$ by any factor that is found on the way. Then $\{d_i\}$ is the set of those j for which this g. c. d. is non-trivial.

REMARK. The greatest common divisor of two polynomials of degree n can be found in $O(n \log^2 n)$ steps, according to the Corollary to [1] Theorem 8.19, but this is using the FFT, and is counting arithmetic operations in a super-ring or pre-image of F_q .

The best g. c. d. algorithm for our purpose may be the naïve one, which requires some $2n^2$ field operations. In our case, we first calculate $g(x) = x^{q^i} \bmod f(x)$, and then find the g. c. d. of $g(x) - x$ and $f(x)$. Since we already know $x^{q^{i-1}} \bmod f(x)$, this requires $\log q$ multiplications and divisions; hence $4 \log q d^2$ field operations. A further $2d^2$ field operations are then required to complete the calculation of the g. c. d. Thus in the worst case, when $f(x)$ is irreducible, one must find the g. c. d. of $f(x)$ with $x^{q^{i-1}} - x$ for $1 \leq i \leq \lceil d/2 \rceil$, so that some $(1 + 2 \log q) d^3$ field operations are required. It is easy to compute α in $O(d^2)$ steps.

We now come to our main result.

ORDER ALGORITHM. *The order of an arbitrary element A of $GL(d, q)$ can be computed using Las Vegas $O(d^3 \log q \log \log q^d)$ finite field operations. This neglects time spent in integer factorisation.*

PROOF. Let f be the minimal polynomial of A , this requires using above lemma on average $O(d^3 \log_q d)$ finite field operations. Computing a multiple n of the order of A requires $O(d^3 \log q)$ finite field operations as described above. The number of distinct primes in n is certainly less than $\log q^d$, therefore the bounded order algorithm requires $O(d \log q \log \log q^d)$ multiplications of group elements provided that the prime factorisations of the $q^i - 1$ that occur in the definition of n are given.

The third idea is as follows. Since the sub- F_q -algebra generated by A is isomorphic to $F_q[x]/(f(x))$, by an isomorphism taking A to $x + (f(x))$, it is sufficient to calculate the multiplicative order of x in the latter ring. This means that multiplications can be done with at worst $O(d^2)$ field multiplications instead of $O(d^3)$ for matrix multiplication. More precisely, the elementary method of multiplying two polynomials of degree less than d , and then dividing the product by $f(x)$ and taking the remainder, requires some $4d^2$ field operations. Note that the p -part of the order of A is known already, and to calculate the p' -part, it is sufficient to calculate the multiplicative order of x in $F_q[x]/(g(x))$, where $g(x)$ is the product of the distinct irreducible factors of $f(x)$. \square

REMARK. If this factorisation is impractical, an estimate of the order of A can be given that is adequate for many purposes.

REMARK. In practice we obtain the precise factorisation of the minimal polynomial which can be found at small extra cost using the Cantor-Zassenhaus method, see [4] page 371. If these irreducible factors are $\{f_i(x) : 1 \leq i \leq u\}$, then to calculate the multiplicative order of x in $F_q[x]/(f(x))$ one needs only to calculate the l. c. m. of the multiplicative orders of x in $F_q[x]/(f_i(x))$.

REMARK. Note that it is possible to multiply two polynomials of degree less than d in less than d^2 operations by various tricks. For example one may use the Fast Fourier Transform, but at the cost of working in a larger ring than F_q . As a simpler alternative, one can multiply two polynomials in $O(d^{\log_2 3})$ field operations by using a divide and conquer trick. Suppose that f and g have degree less than $2n$, and write $f = x^n a + b$, and $g = x^n c + d$, where a, b, c, d are polynomials of degree less than n . Then

$$f \times g = x^{2n} ac + x^n((a+b)(c+d) - ac - bd) + bd$$

and now the three products $ac, bd, (a+b)(c+d)$ can be calculated recursively by the same algorithm, taking n to be a power of 2 so that odd degrees will not arise until the last step. This clearly gives the stated complexity. Using the fast g. c. d. algorithm in [1] one now calculates the g. c. d. of two polynomials of degree less than d in $O(d^{\log_2 3} \log d)$ field operations, and the complexity of finding the remainder after dividing one polynomial of degree less than d by another has the same complexity, up to a constant, as multiplication, so the bounded order algorithm can now be executed in $O(d^{1+\log_2 3} \log d \log q \log \log q^d)$ finite field operations; but it may be that difficulties with factorising $q^i - 1$ for $1 \leq i \leq d$ dominate before these tricks pay off.

REMARK. Integers of the form $p^n - 1$ are easier to factorise than arbitrary integers. The most obvious idea is first to split the cyclotomic polynomial $x^n - 1$ into irreducible factors, and then substitute p for x and find the prime factorisation of the factors that arise. We have also used the list of factorisations in [2] which is available from <ftp://ftp.ox.ac.uk/pub/math/cunningham>.

3. Timings

The algorithm has been programmed in the group-theoretic language GAP [7], using the straightforward $O(d^2)$ method of multiplying polynomials. The following timings were obtained running the program on an Intel Pentium P5, 90 Mhz, running FreeBSD 2.0.5. In the first table we consider elements of groups whose elements have relatively small orders. The timings are the average for ten random elements. The columns are indexed by the groups. The first row gives the dimension of the representation, the second row, labelled "new", gives the timing in seconds for our algorithm. The third row, labelled "old", gives the timing for the previously used algorithm that computes the order by repeatedly spinning and powering. The second table gives times for random elements of the general linear groups in the same dimensions. In all cases in these tables, $q = 7$. Since the spinning and powering algorithm requires $O(d^2 q^d)$ finite field operations to compute the order of a random element of $GL(d, q)$, it would not be feasible to use this method in this case.

SMALL ORDERS

	$L(2, 13)$	M_{22}	$He.2$	M_{22}
d	14	21	50	154
new	0.075	0.10	0.40	2.20
old	0.007	0.03	0.50	4.00

LARGE ORDERS

	$GL(14, 7)$	$GL(21, 7)$	$GL(50, 7)$	$GL(154, 7)$
d	14	21	50	154
new	0.010	0.018	1.50	23

As a larger example, we applied the algorithm to an element of $GL(750, 2)$, and the answer $2 \cdot 8191 \cdot (2^{735} - 1)$ was returned after 1250 seconds. 1% was spent calculating the minimal polynomial, 9% calculating the distinct degree factorisation, and 90% in the bounded order algorithm.

4. Variations

1. It is a triviality to adapt the algorithm to calculate the order of a matrix modulo the scalar matrices. If the f_i are the irreducible factors of the minimal polynomial of a given matrix A , we first compute the projective order n_i of x in $F_q[x]/(f_i(x))$ and use the bounded order algorithm again to find the smallest positive integers n dividing $q - 1$ such that all remainders of $x^{n_i n}$ modulo $f_i(x)$ are equal. The p' -part of the projective order is the l. c. m. of the $n_i n$, and the p -part of the order and the projective order coincide. This version is used in our algorithm for determining whether a given finite subset of $GL(d, q)$ generates a subgroup that contains the special linear group; see [3].

2. If the p' -part of the order of the element A is all that is required, one can use the characteristic polynomial instead of the minimal polynomial; more precisely, replace the minimal polynomial by the product of the distinct prime factors of the characteristic polynomial. This has the advantage that the characteristic polynomial can always be computed in $O(d^3)$ field operations.

3. It is often useful to find elements of given prime or prime-power order. The obvious way to do this is to find a random element in the group, and if it is of order a multiple of the required order then raise it to a suitable power. It is much more efficient, however, to truncate the order algorithm when it is clear that the given element does not have a suitable order. For example, an element cannot have order a multiple of the characteristic of the underlying field if the minimal polynomial does not have a repeated root. It cannot have an order that is a multiple of some prime power unless that prime power divides $q^i - 1$, where i is the degree of some irreducible factor of the minimal polynomial of A .

4. The algorithm of Niemeyer and Praeger for recognising classical groups [6] requires one to determine whether or not an element of $GL(d, q)$ is a ppd element; that is to say, an element of order n where n is a multiple of some prime ℓ that

divides $q^e - 1$ for some e where $d/2 < e \leq d$, but $l \nmid p^t - 1$ for any $t < \log_p q^e$. If A is a ppd element it is also important to know whether n, l satisfy the auxiliary condition $l > e + 1$ or $(e + 1)^2 | n$. These questions can be answered by using a simplified version of our order algorithm. First compute the characteristic polynomial $f(x)$ of G . If $f(x)$ has no irreducible factor of degree e greater than $d/2$ then clearly A is not a ppd element. If it does contain such a factor $h(x)$ we compute in $F_q[x]/(h(x))$. We produce a factorisation of $q^e - 1$ as $n_1 n_2 n_3$ where n_1 is a product of primes that divide $p^t - 1$ for some $t < \log_p q^e$, and n_2 is $e + 1$ if this is a prime dividing n exactly once, and is 1 otherwise. Now compute x^{n_1} and $x^{n_1 n_2}$. If the latter is not 1 then A is a ppd element that also satisfies the auxiliary condition. If the former is 1 then A is not a ppd element. To compute the characteristic polynomial of A , and to determine the factor $h(x)$, if this exists, require $O(d^3)$ and $O(d^3 \log q)$ field operations respectively. An upper bound for the number of bit operations in the factorisation is $O(d^3 \log d \log^3 q)$. However, the time spent in the integer factorisation is a negligible proportion of the total time required.

We are grateful to E. O'Brien, L. H. Soicher and the late M. R. B. Clark for valuable suggestions.

References

- [1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison Wesley, Massachusetts, London, Amsterdam, Ontario, Sydney, 1974.
- [2] J. Brillhart, D. H. Lehmer, J. Selfridge, S. S. Wagstaff Jr., and B. Tuckerman, *Factorizations of $b^n \pm 1$, $b = 2, 3, 5, 6, 7, 10, 11, 12$ up to high powers*, Contemporary Mathematics **22** (1983), Amer. Math. Soc., Providence, RI; 2nd. edition, 1988.
- [3] Frank Celler and C. R. Leedham-Green, *A Non-Constructive Recognition Algorithm for the Special Linear and Other Classical Groups*, To appear in the DIMACS proceedings.
- [4] K. O. Geddes, S. R. Czapora, and G. Labahn, *Algorithms for Computer Algebra*, Kluwer Academic Publishers, Dordrecht, 1992.
- [5] D. E. Knuth, *The Art of Computer Programming, Vol. 2*, Addison Wesley, Massachusetts, London, Amsterdam, Ontario, Sydney, 1981.
- [6] Alice C. Niemeyer and Cheryl E. Praeger, *A Recognition Algorithm for Classical Groups over Finite Fields*, In preparation.
- [7] Martin Schönert *et al.*, *GAP—Groups, Algorithms and Programming*, Lehrstuhl D für Mathematik, RWTH Aachen, 1994.

LEHRSTUHL D FÜR MATHEMATIK, RWTH-AACHEN, 52062 AACHEN, GERMANY
E-mail address: Frank.Celler@Math.RWTH-Aachen.DE

SCHOOL OF MATHEMATICAL SCIENCES, QUEEN MARY AND WESTFIELD COLLEGE, UNIVERSITY OF LONDON, MILE END ROAD, LONDON E1 4NS, GREAT BRITAIN