MATHEMATICS OF COMPUTATION
VOLUME 39, NUMBER 160
OCTOBER 1982, PAGES 663–670

Computing in Permutation and Matrix Groups I: Normal Closure, Commutator Subgroups, Series*

By Gregory Butler and John J. Cannon

Abstract. This paper is the first in a series which discusses computation in permutation and matrix groups of very large order. The fundamental concepts are defined, and some algorithms which perform elementary operations are presented. Algorithms to compute normal closures, commutator subgroups, derived series, lower central series, and upper central series are presented.

1. Introduction. Our aim in this paper is to present efficient methods for computing the derived and lower central series of a group G of permutations or matrices over a finite field. The methods combine the original algorithms of Cannon [3], which are independent of the group representation, with the techniques of Sims [7], [8] and Butler [1] for computing in permutation and matrix groups. The resultant algorithms have been implemented as part of the group theory system CAYLEY [4]. Our experience with these implementations suggests that the algorithms are applicable to groups to very large order—at least 10^{16}—and to permutation groups of degree up to 1000 in some instances. For the algorithms to be applicable to a matrix group, the group must be almost faithful on a small orbit (or union of orbits) of the one-dimensional subspaces. That is, the kernel of the transitive constituent homomorphism should be very small, and the length of the orbit should be less than 1000 or so.

We begin in Section 2 by reviewing the general algorithms for computing normal closures, commutator subgroups, the derived and lower central series. A new algorithm to compute the upper central series is presented in Section 3. The critical operations of these algorithms are testing whether an element g of G is in a subgroup H, replacing H by $\langle H, g\rangle$, and comparing the orders of two subgroups. These operations can be efficiently performed in permutation and matrix groups using the concepts presented in Section 4 and the algorithms presented in Sections 5 and 6. We conclude in Section 7 with some examples of our experience with the implementations in CAYLEY.

The notation used is:

identity	identity element of a group
x^g	the image of x under g
h^g	$g^{-1}hg$
$[g, h]$	$g^{-1}h^{-1}gh$

Received May 12, 1980; revised March 24, 1982.
1980 *Mathematics Subject Classification.* Primary 20–04, 20G40, 20F14.
Key words and phrases. Algorithm, permutation group, matrix group.
*This work was partially supported by the Australian Research Grants Committee.

0025-5718/82/0000-0372/$02.75

$[G, H]$	the group generated by $[g, h]$ for all $g \in G, h \in H$.
$\langle g_1,\ldots,g_m\rangle$	the group generated by $g_1,\ldots,g_m$
$a \leftarrow b$	a is assigned the value b

2. Normal Closure, Commutator Subgroup, Derived and Lower Central Series. Let H be a subgroup of a group G. The normal closure $\mathrm{ncl}_G(H)$ of H in G is the smallest normal subgroup of G which contains H. Therefore $\mathrm{ncl}_G(H) = \langle H^g \mid g \in G\rangle$. Given a set of generators of H and a set of generators of G, we compute the normal closure by extending H by sufficient conjugates of the generators of H so that the resultant subgroup is normal in G. More precisely, suppose that $G = \langle g_1,\ldots,g_m\rangle$ and $H = \langle h_1,\ldots,h_l\rangle$, then $\mathrm{ncl}_G(H)$ is computed as follows:

ALGORITHM NCL

NCL1: [initialize]
$K \leftarrow H, k \leftarrow l, i \leftarrow 0.$

NCL2: [choose next generator of K to conjugate]
$i \leftarrow i + 1.$
if $i > k$ then $\mathrm{ncl}_G(H) \leftarrow K$ and stop.
$j \leftarrow 0.$

NCL3: [test if $h_i^{g_j}$ is in K for each j]
$j \leftarrow j + 1.$
if $j > m$ then go to (2).
$g \leftarrow h_i^{g_j}.$
if $g \in K$ then go to (3).

NCL4: [conjugate of h_i is not in K so extend K]
$k \leftarrow k + 1, h_k \leftarrow g, K \leftarrow \langle K, h_k\rangle.$
go to (3).

When $H_1 = \langle a_1,\ldots,a_m\rangle$ and $H_2 = \langle b_1,\ldots,b_l\rangle$ are normal subgroups of G, the commutator subgroup $[H_1, H_2]$ is the normal closure in G of $H = \langle [a_i, b_j] \mid i = 1,\ldots,m, j = 1,\ldots,l\rangle$. We can test whether $[a_i, b_j]$ is in the group generated by the previous commutators. If it is, then we discard it from the generating set of H. If it is not, then we extend the group generated by the previous commutators. Once H is computed, the commutator subgroup $[H_1, H_2]$ is computed by Algorithm NCL.

As the derived and lower central series can be defined in terms of commutator subgroups, the method of their computation is now straightforward. The termination of the series can be decided by the comparison of the orders of two subgroups—the last term in the series and the subgroup which is potentially the next term in the series.

3. Upper Central Series. The upper central series of a group G is the sequence $\{\text{identity}\} = G^0 \leqslant G^1 \leqslant G^2 \leqslant \cdots$ of subgroups where G^i, $i \geqslant 1$, is the inverse image (under the canonical homomorphism) in G of $Z(G/G^{i-1})$. Suppose $G = \langle g_1,\ldots,g_m\rangle$ and that $\{r_1,\ldots,r_l\}$ is a set of representatives of the conjugacy classes of elements of G. Then $G^i = \langle G^{i-1}, \{r_j \mid [r_j, g_k] \in G^{i-1} \text{ for all } k\}\rangle$ for $i \geqslant 1$. Given a set of class representatives, the following algorithm uses this characterization of G^i to compute the upper central series of G.

ALGORITHM UCS

UCS1: [initialize]
$i \leftarrow 0$, $H \leftarrow \{\text{identity}\}$, $G^0 \leftarrow \{\text{identity}\}$.

UCS2: [begin the next subgroup in the series]
$i \leftarrow i + 1$, $j \leftarrow 0$.

UCS3: [is r_j a generator of the next subgroup]
$j \leftarrow j + 1$.
if $j > l$ then go to (6).
if $r_j \in H$ then go to (3).
$k \leftarrow 0$.

UCS4: [test $[r_j, g_k] \in G^{i-1}$ for all k]
$k \leftarrow k + 1$.
if $k > m$ then go to (5).
$g \leftarrow [r_j, g_k]$.
if $g \notin G^{i-1}$ then go to (3).
go to (4).

UCS5: [$G^{i-1} \leqslant H \leqslant G^i$]
$H \leftarrow \langle H, r_j \rangle$.
go to (3).

UCS6: [next subgroup is complete]
if $|H| = |G^{i-1}|$ then stop.
$G^i \leftarrow H$.
if $|G^i| = |G|$ then stop else go to (2).

The computation of conjugacy classes of elements is with present techniques restricted to groups of order less than 10^4, so the range of application of Algorithm UCS is quite restricted.

4. Concepts. Let G be a group acting faithfully on the finite set X. For a group of permutations of Ω we take $X = \Omega$, and for a group of matrices acting on the vector space V over a finite field we take $X = V \cup W$, where W is the set of all one-dimensional subspaces of V.

Suppose a subset $S = \{s_1, \ldots, s_m\}$ of G is given and let $H = \langle S \rangle$. For $x \in X$, the *orbit* x^H of x under H is the set of images x^g as g runs over H. The *stabilizer* H_x of x in H is the set of elements of H which fix x. For $y \in x^H$, choose an element $u(y)$ of H which maps x to y. Then $U = \{u(y) \mid y \in x^H\}$ is a set of right coset representatives for H_x in H. A *Schreier vector* for x^H is a map v: $x^H \setminus \{x\} \to S$ such that, for each $y \in X^H \setminus \{x\}$, there is a unique sequence $[y_1, \ldots, y_r]$ with $y_i \in x^H \setminus \{x\}$, $y_r = y$, and such that $v(y_1)v(y_2) \cdots v(y_r)$ maps x to y. The map w: $x^H \setminus \{x\} \to x^H$, where $w(y) = y^{v(y)^{-1}}$ is called a vector of backward pointers. The Schreier vectors and the vectors of backward pointers offer a space-efficient method of storing U, and are implemented as maps v: $X \to \{-1, 0, 1, 2, \ldots, m\}$ for permutation groups, v: $x^H \to \{-1, 1, 2, \ldots, m\}$ for matrix groups, and w: $\{1, 2, \ldots, |x^H|\} \to \{0, 1, 2, \ldots, |x^H|\}$ for matrix and permutation groups. In the implementation of the vector w there is an implicit order on the orbit x^H. This order is given by the order in which the members of the orbit are generated. See Algorithm ORB of Section 5 for more details.

A sequence $B = [x_1,\ldots,x_k]$ of elements of X is a *base* for H if no nonidentity element of H fixes B pointwise. For $i = 1,2,\ldots,k+1$, let $H^{(i)} = H_{x_1,x_2,\ldots,x_{i-1}}$. Thus $H = H^{(1)} \geqslant H^{(2)} \geqslant \cdots \geqslant H^{(k+1)} = \{\text{identity}\}$. For $i = 1,2,\ldots,k$, let $\Delta_i = x_i^{H^{(i)}}$, and let U_i be a set of right coset representatives for $H^{(i+1)}$ in $H^{(i)}$. We call Δ_i and U_i the *ith basic orbit* and *ith basic transversal*, respectively. Each element g of H is uniquely expressible as a product $u_k \cdots u_1$ with $u_i \in U_i$, for all i. Hence $|H| = \prod_{i=1}^{k} |\Delta_i| = \prod_{i=1}^{k} |U_i|$.

A set of elements of H is a *strong generating* set of H relative to B if, for $i = 1,2,\ldots k$, $S \cap H^{(i)}$—which we will call $S^{(i)}$—is a generating set of $H^{(i)}$. If a base and strong generating set of H are known, then Algorithm ORB of Section 5 can be used to compute Δ_i and U_i, for each i. The characterization of the elements of H as products $u_k \cdots u_1$ means that H can be described quite compactly by a base B, a strong generating set S, and a Schreier vector v_i and a vector w_i of backward pointers of each orbit Δ_i, $i = 1,2,\ldots,k$. Another advantage of knowing a base for a group is that relations amongst elements of the group need only be verified on the base points. This is used in Algorithms INSV and XSM to improve efficiency.

5. Fundamental Algorithms. For completeness we present two fundamental algorithms. Both are well known. See, for example, [6], [7].

Given a set $S = \{s_1,\ldots,s_m\}$ of generators of H, the following algorithm computes the orbit Y of x, a Schreier vector v, the vector w of backward pointers, and a set $U = \{u(y) \mid y \in Y\}$ of right coset representatives for H_x in H. Basically the algorithm adds the image y^{s_t} of a point y of Y under a generator s_t until Y is closed under the action of H.

ALGORITHM ORB

ORB1: [initialize]
$r \leftarrow 1, y_r \leftarrow x, j \leftarrow 0.$
$v(x) \leftarrow -1, w(x) \leftarrow 0, u(x) \leftarrow$ identity.

ORB2: [choose jth element of orbit to act on]
$j \leftarrow j + 1.$
if $j > r$ then $Y \leftarrow \{y_1,\ldots,y_r\}$ and stop.
$t \leftarrow 0, y \leftarrow y_j.$

ORB3: [choose tth generator to act by]
$t \leftarrow t + 1.$
if $t > m$ then go to (2).

ORB4: [act on jth element by tth generator]
$z \leftarrow y^{s_t}.$
if z is in $\{y_1,\ldots,y_r\}$ then go to (3).
$r \leftarrow r + 1, y_r \leftarrow z, v(z) \leftarrow t, w(z) \leftarrow j, u(z) \leftarrow u(y)s_t.$
go to (3).

For the purposes of Algorithm XSM we require a variation of this algorithm which, given an orbit Y, a Schreier vector v, and a vector w of backward pointers relative to the group $(s_1, s_2,\ldots,s_m)$, extends Y, v and w so they correspond to the extended group $(s_1, s_2,\ldots,s_m, s_{m+1})$. The necessary changes to Algorithm ORB are straightforward.

The next algorithm performs the first of the critical operations for computing series. It determines whether an element g of G is in H by attempting to express g as a product $u_k \cdots u_1$ of coset representatives. It assumes that a description of H (by a base, etc., as at the end of Section 4) is known and that a base C for G is known.

ALGORITHM INSV

INSV1: [initialize]
$i \leftarrow 0$, $D \leftarrow B^g$, $E \leftarrow C^g$.

INSV2: [$D = [d_1, \ldots, d_k]$. Next level of base]
$i \leftarrow i + 1$.
if d_i is not in Δ_i then g is not in H.

INSV3: [determine u_i from Schreier vector. Act on D, E by u_i^{-1}]
$j \leftarrow v_i(d_i)$.
if $j = -1$ then go to (4).
$D \leftarrow D^{s_j^{-1}}$, $E \leftarrow E^{s_j^{-1}}$.
go to (3).

INSV4: if $i < k$ then go to (2).
g is in H if and only if $E = C$.

The frequent use of the inverses of the strong generators in Algorithm INSV is the reason the inverses are also stored. In the language of Sims and Leon, S is assumed to be closed under inverses.

6. The Extending Schreier Method. In 1967 Sims [7] developed the Schreier method to compute a base and strong generating set of a permutation group. It is founded on Schreier's lemma which states that, since $U = \{u(y) \mid y \in x^G\}$ is a set of coset representatives for H_x in H, the set $\{u(y)su(y^s)^{-1} \mid y \in x^H, s \in$ generating set of $H\}$ of Schreier generators generate H_x. The Schreier method was generalized to matrix groups in Butler [1] and variations employing coset enumeration have also been developed (Sims [10], [11], Leon [6], Butler [2]). Here we present a version particularly applicable to the algorithms of Sections 2 and 3. Since the details are important to the efficiency of the Schreier method, we will be quite explicit. It is perhaps useful for the reader to first become familiar with the description of the Schreier method in [1].

The algorithm assumes that a base C for G is known, and that a base $B = [x_1, \ldots, x_k]$ and a strong generating set $S = \{s_1, \ldots, s_m\}$ of H are known. It is further assumed that every element in B is also in C and that S is closed under inverses. For $i = 1, 2, \ldots, k$, the Schreier vector v_i and the vector w_i of backward pointers of $H^{(i)}$ are known, thus enabling the coset representatives $u_i(y)$ for $H^{(i+1)}$ in $H^{(i)}$ to be expressed as words in the elements of S, and enabling the "stripping" procedure of step XSM4 to be performed as in Algorithm INSV. By representing an element as a word and operating on the base instead of the whole of X, we produce the bulk of the gains of Algorithm XSM over the Schreier method. The remainder comes by avoiding the repeated consideration of the same Schreier generator each time a level is revisited. This is achieved through the use of Γ_i which stores those points y for which it is known that $u_i(y)su_i(y^s)^{-1}$ is in $\langle S^{(i+1)} \rangle$ for all generators s of $S^{(i)}$ except possibly the last. We let gen$[i] = \{j \mid s_j \in S^{(i)}\}$, and we denote the jth

element of gen[i] by gen[i, j]. The set Y_i consists of those elements of Δ_i whose corresponding Schreier generators have been or are being processed during the current visit to level i.

The algorithm extends the base and strong generating set of H to a base and strong generating set of $\langle H, g\rangle$, where $g \in G \setminus H$. Algorithm XSM also computes the order of the subgroup thus performing the second and third operations which are critical to the efficiency and range of applicability of the algorithms of Sections 2 and 3.

ALGORITHM XSM

XSM1: [initialize]
for $i = 1, 2, \ldots, k$
$$\Gamma_i \leftarrow \Delta_i,\ Y_i \leftarrow \varnothing,\ \text{gen}[i] \leftarrow \{j \mid s_j \in S^{(i)}\}.$$
$h \leftarrow g$, $i \leftarrow 0$.
let $l \leqslant k + 1$ be the greatest integer such that
$$[x_1, \ldots, x_{l-1}]^h = [x_1, \ldots, x_{l-1}].$$
if $l \leqslant k$ then go to (7).
go to (6).

XSM2: [next coset representative. Act only by last generator if $y_i \in \Gamma_i$]
if $Y_i = \Delta_i$ then go to (10).
choose $y_i \in \Delta_i \setminus Y_i$.
$Y_i \leftarrow Y_i \cup \{y_i\}$.
if $y_i \in \Gamma_i$ then $r_i \leftarrow |\text{gen}[i]| - 1$ else $r_i \leftarrow 0$.

XSM3: [next generator of $S^{(i)}$ for Schreier generator]
$r_i \leftarrow r_i + 1$.
if $r_i > |\text{gen}[i]|$ then go to (2).
form $h = u_i(y_i)s_{\text{gen}[i,r_i]}$ as a word.
$D \leftarrow C^h$, $l \leftarrow i - 1$.

XSM4: ["strip" h to test if Schreier generator is redundant]
$l \leftarrow l + 1$.
if $l > k$ then go to (5).
$y \leftarrow x_l^h$.
if $y \notin \Delta_l$ then go to (7).
$h \leftarrow hu_l(y)^{-i}$ as a word, $D \leftarrow D^{u_l(y)^{-1}}$.
go to (4).

XSM5: if $D = C$ then go to (3).

XSM6: [h fixes B so extend base]
$k \leftarrow k + 1$, choose $x_k \in C \setminus B$ moved by h.

XSM7: [new strong generator]
$m \leftarrow m + 1$, $s_m \leftarrow h$ (evaluated as an element), $S \leftarrow S \cup \{s_m, s_m^{-1}\}$.

XSM8: [s_m belongs to $S^{(i+1)}, \ldots, S^{(l)}$]
$i \leftarrow i + 1$.
if $i > l$ then go to (9).
$Y_i \leftarrow \varnothing$, gen$[i, |\text{gen}[i]| + 1] \leftarrow m$.
extend Δ_i, v_i, w_i.
go to (8).

XSM9: [continue at level l]
$i \leftarrow l$, go to (2).
XSM10: [know a base and strong generating set for $\langle S^{(i)} \rangle$]
$\Gamma_i \leftarrow \Delta_i$.
$i \leftarrow i - 1$.
if $i = 0$ then stop.
if $Y_i = \varnothing$ then go to (2) else go to (3).

Two remarks concerning Algorithm XSM are in order. Firstly, the extending generator g is usually "stripped" as in step XSM4, although this is not necessary for the correctness of the algorithm. Secondly, there is a theoretical reason for extending the orbits and Schreier vectors in step XSM8. The proof of Schreier's lemma assumes that for a fixed set of coset representatives each of the Schreier generators is considered. As we wish to avoid reconsidering those Schreier generators previously examined, we require that the set of coset representatives be extended rather than recomputed upon construction of a new strong generator.

For permutation groups with short base C the algorithm is very fast. For example, each application of Algorithm XSM in Section 7 (1) took an average 1.4 seconds on a CDC Cyber 170/730, and it has been used successfully on groups of degree 2000 with bases of length 3 to 10. For matrix groups where $\dim V$ is small, the length of the base is usually larger than $\dim V$, and it is more efficient not to represent elements as words and not act on just the base. Each application of Algorithm XSM in Section 7 (2) took an average of 20 seconds on a CDC Cyber 170/730.

7. Conclusion. We consider some applications of the methods. All times are for a CDC Cyber 170/730.

(1) The Fibonacci group $F(2, 9)$ has a homomorph of order $2^3 \cdot 5^{18} \cdot 19$ with a faithful permutation representation of degree 190 (Havas, Richardson and Stirling [5]) with a base of length 19. It took 91 seconds to compute the three distinct terms of the lower central series. They have order $2^3 \cdot 5^{18} \cdot 19$, $2 \cdot 5^{18}$, and 5^{18}. Algorithm XSM was applied 55 times.

(2) The group $G = \langle a, b, c \rangle$ of 7×7 matrices over $GF(5)$ (Sims [9]) has order $2^5 \cdot 3 \cdot 5^6$. It took 102 seconds to compute the derived subgroup of order $2^3 \cdot 3 \cdot 5^6$. Algorithm XSM was applied 5 times.

(3) A Sylow 2-subgroup G of $GL(5, 2)$ has order 2^{10}. Regarded as a group of 5×5 matrices over $GF(2)$, the computation of the upper central series took 41 seconds. The terms have order 1, 2, 2^3, 2^6 and 2^{10}. The 10 applications of Algorithm XSM took 5 seconds. Regarded as a permutation group of degree 31 with a base of length 5, the times were 5 seconds and 2 seconds, respectively. In both cases the computation of the lower central series required approximately one-third of the time.

In conclusion it is clear that the combination of the algorithms for groups with an arbitrary representation and the powerful techniques for permutation and matrix groups has greatly increased the range of applicability of the algorithms. The extent of their applicability is indicated by the above examples.

Department of Mathematics
McGill University
Montreal, Quebec, Canada H3A 2K6

Department of Computer Science
Concordia University
Montreal, Quebec, Canada H3G 1M8

Department of Pure Mathematics
University of Sydney
Sydney, Australia

1. GREGORY BUTLER, "The Schreier algorithm for matrix groups," *SYMSAC'76* (Proc. 1976 A. C. M. Sympos. on Symbolic and Algebraic Computation, Yorktown Heights, N. Y., 1976), R. D. Jenks (ed.), A. C. M., New York, 1976, pp. 167–170.

2. GREGORY BUTLER, *Computational Approaches to Certain Problems in the Theory of Finite Groups*, Ph. D. Thesis, University of Sydney, 1979.

3. JOHN J. CANNON, "Computing local structure of large finite groups," *Computers in Algebra and Number Theory* (Proc. Sympos. on Applied Mathematics, New York, 1970), G. Birkhoff and M. Hall, Jr. (eds.), SIAM—AMS Proceedings, vol. 4, Amer. Math. Soc., Providence, R. I., 1971, pp. 161–176.

4. JOHN J. CANNON, "Software tools for group theory," *Proc. Sympos. Pure Math.*, vol. 37, Amer. Math. Soc., Providence, R. I., 1980, pp. 495–502.

5. GEORGE HAVAS, J. S. RICHARDSON & LEON S. STERLING, "The last of the Fibonacci groups," *Proc. Roy. Soc. Edinburgh Sect. A*, v. 83, 1979, pp. 199–203.

6. JEFFREY S. LEON, "On an algorithm for finding a base and strong generating set for a group given by generating permutations," *Math. Comp.*, v. 35, 1980, pp. 941–974.

7. CHARLES C. SIMS, "Computational methods for permutation groups," *Computational Problems in Abstract Algebra* (Proc. Conf., Oxford, 1967), J. Leech (ed.), Pergamon Press, Oxford, 1970, pp. 169–183.

8. CHARLES C. SIMS, "Determining the conjugacy classes of a permutation group," *Computers in Algebra and Number Theory* (Proc. Sympos. on Applied Mathematics, New York, 1970), G. Birkhoff and M. Hall, Jr. (eds.), SIAM—AMS Proceedings, vol. 4, Amer. Math. Soc., Providence, R. I., 1971, pp. 191–195.

9. CHARLES C. SIMS, "The existence and uniqueness of Lyons' group," *Finite Groups'72* (Proc. Gainesville Conference, Gainesville, Florida, 1972), T. Gagen, M. P. Hale, and E. E. Shult (eds.), North-Holland, Amsterdam, 1973, pp. 138–141.

10. CHARLES C. SIMS, "Some algorithms based on coset enumeration," 1974. (Manuscript.)

11. CHARLES C. SIMS, "Some group-theoretic algorithms," *Topics in Algebra* (Proc. 18th Summer Research Institute of the Australian Math. Soc., A. N. U., Jan. 9—Feb. 17, 1978), M. F. Newman (ed.), Lecture Notes in Math., vol. 697, Springer-Verlag, Berlin and New York, 1978, pp. 108–124.