Automorphism group computation and
isomorphism testing in finite groups

JOHN J. CANNON\textsuperscript{1} AND DEREK F. HOLT\textsuperscript{2}

\textsuperscript{1}School of Mathematics, University of Sydney, Sydney, Australia
\textsuperscript{2}Mathematics Institute, University of Warwick, Coventry, Great Britain

Abstract
A new method for computing the automorphism group of a finite permutation group and for testing two such groups for isomorphism is described. Some performance statistics are included for an implementation of these algorithms in the \texttt{Magma} language.

1. Introduction
In this paper, we describe a new method for efficiently computing the automorphism group of a finite group $G$, and for testing two finite groups $G$ and $H$ for isomorphism. It has been implemented for finite permutation groups within the language of the \texttt{Magma} computational algebra system [Bosma et al., 1997, Bosma and Cannon, 2002b], and its performance is very encouraging. These algorithms perform effectively on the majority of groups of order up to a million that are commonly encountered, and also on many much larger groups. There are some types of groups, principally groups of prime power order, on which they do not perform well, but satisfactory alternative methods are already available for many such examples.

Since it is not always possible (or easy) to find a reasonably low degree permutation representation of $\text{Aut}(G)$, we return the result as a finitely presented group. The user can then either use this presentation to search for a more convenient representation of $\text{Aut}(G)$, or alternatively use an auxiliary function, which attempts to find a low degree faithful permutation representation of $\text{Aut}(G)$ on a union of conjugacy classes of $G$.

While the current implementation is for permutation groups, the methods are generic, and could be carried out equally well in any category of finite groups for which basic algorithms for structural computation were available. It is, however, essential to be able to compute efficiently with automorphisms of $G$. That is, it must be possible to define an automorphism $\alpha$ by specifying the images under

\texttt{Magma} language.
of a given generating set of $G$, to compute the image of an arbitrary element of $G$ under $\alpha$, and to compose and invert elements of Aut($G$). Methods for the case when $G$ is a finite permutation group are summarized in section 2.3 below.

Prior to the algorithms described in this paper, the most recent general purpose procedure for computing the automorphism group of a finite group $G$ performs a backtrack search through the possible images of a fixed generating set for $G$ under automorphisms. The automorphism group is represented as a permutation group acting on a union of some of the conjugacy classes of $G$. Consequently, some of the techniques developed by Sims for backtrack searches in permutation groups may be applied. This method is described in detail in Robertz [1976] and is used in GAP [Schönert et al., 1995] and in CAYLEY [Cannon, 1984]. The procedure is quite satisfactory for small groups, but its effectiveness diminishes rapidly and becomes less predictable with increasing order. An earlier method due to V. Felsch and Neubüser which exploits detailed structural information of $G$ is described in Felsch and Neubüser [1968] and Felsch and Neubüser [1970].

With the recent development of very efficient algorithms for computing detailed structural information for groups, it seems reasonable to make use of this structure when calculating their automorphism groups and testing them for isomorphism. There is a general approach to group-theoretical calculations that has proved particularly useful in the special case of soluble groups $G$ defined by power-conjugate presentations. This is to find a series of normal subgroups with elementary abelian layers, and to solve the problem in the associated factor groups of $G$, starting with the top factor and lifting through each layer in turn, until we finally have solved the problem for $G$ itself. In particular, such methods for computing automorphism groups of finite $p$-groups [O’Brien, 1995, Eick et al., 2002] and soluble groups [Smith, 1995] have been developed and implemented. Both algorithms are available in Magma (the version for soluble groups was implemented by Mike Slattery), and both are also available as GAP share packages (AutPGrp and autag, respectively).

The idea of applying essentially the same strategy to a general finite group was used in the algorithm described in Cannon et al. [2001] for computing the subgroup lattice of a finite group. Of course, a general group does not necessarily have elementary abelian chief factors; our strategy is (roughly) to push the non-abelian part of the group to the top of the group and to handle it independently in the first stage of the calculation.

The remainder of the paper is organized as follows. Section 2 presents an overview of the automorphism group and isomorphism testing algorithms. Section 3 provides details of these algorithms for groups with trivial Fitting subgroup. In section 4, we describe how to find a suitable series of characteristic subgroups of $G$, together with a presentation of $G$ that ‘exhibits the series’ in a sense that will be defined in subsection 2.3 below. In section 5, we present details of the lifting process of computing Aut($G/N$) from Aut($G/M$), where $M$ and $N$ are characteristic subgroups of $G$, and $M/N$ is elementary abelian. In section 6, we briefly describe some features of the Magma implementation,
in terms of the information returned and related facilities for computing with automorphism groups. We also provide some performance statistics which help to give an idea of the range of applicability of the implementation in its current state.

We are grateful to unnamed referees for a careful reading of the paper, and for many helpful comments.

2. Overview of the Algorithms

2.1. The Automorphism Group Algorithm

We start by finding a series of characteristic subgroups

\[ 1 = N_r < N_{r-1} < \ldots < N_1 = L < G \]

of our given group \( G \), such that each \( N_i/N_{i+1} \) is elementary abelian and \( G/L \) has no non-trivial soluble normal subgroup. We shall call groups with this last property trivial-Fitting groups, since they have trivial Fitting subgroups. Of course, \( L \) is uniquely defined as the largest normal soluble subgroup of \( G \), and \( G/L = 1 \) if \( G \) itself is soluble.

A trivial-Fitting group \( G \) has a socle \( S \) which is a direct product of nonabelian simple groups, the socle factors \( S_i \), and these factors are permuted under conjugation in the group. In fact \( G \) can be embedded in a direct product \( X \) of wreath products of the form \( \text{Aut}(S_i) \wr \text{Sym}(d_i) \), where we include one such wreath product for each isomorphism type of the socle factors \( S_i \), and \( S_i \) occurs \( d_i \) times as a factor of \( S \). Then \( \text{Aut}(G) \) may be calculated as the normalizer of the image of \( G \) in \( X \). The \( \text{Aut}(S_i) \) still need to be computed, but these considerations effectively reduce the computation in the trivial-Fitting case to the case when \( G \) is simple. Much of the necessary information in the simple case can be found in the ATLAS [Conway et al., 1985]. Our approach is to store this information for each individual isomorphism type of nonabelian simple group, and we have currently done this for all such groups of order up to 16,482,816 and for some larger groups, including all alternating groups up to degree 50. The information may be accessed as required in a specific automorphism group calculation. The same method was applied for the subgroup lattice calculation described in Cannon et al. [2001], but the storage requirement for automorphism groups is considerably smaller. For detailed information on the simple groups stored in Magma, see Bosma and Cannon [2002a].

Once \( \text{Aut}(G/L) = \text{Aut}(G/N_1) \) has been found, we proceed by a lifting process through the elementary abelian layers, computing \( \text{Aut}(G/N_i) \) successively for \( i = 2, \ldots, r \). This is the same general strategy that is employed by Smith [1995] for soluble groups, although we have introduced a few refinements. For example, in order to return a group rather than just a sequence of generating automorphisms, we return a presentation with generators and defining relators of \( \text{Aut}(G) \). This presentation has the property that an initial subsequence of its defining generators generates the inner automorphism group of \( G \).
For the main inductive step in the process, suppose that $G$ has characteristic subgroups $N$ and $M$, such that $N \leq M$ and $M/N$ is an elementary abelian $p$-group of order $p^d$ for some prime $p$ and some integer $d > 0$, and that $A_M = \text{Aut}(G/M)$ has already been calculated. The section $M/N$ can be regarded as a $KG/M$-module, where $K = \text{GF}(p)$ is the finite field of order $p$. Since the subgroups are characteristic, all automorphisms of $G$ fix both $M$ and $N$. Then $A_N = \text{Aut}(G/N)$ has normal subgroups $B$ and $C$ with $C \leq B$, where $B$ consists of those automorphisms that induce the identity on $G/M$ and $C$ consists of those which, in addition, induce the identity on $M/N$. The elements of $C$ correspond to derivations from $G/M$ to the module $M/N$, and are easily computed. Elements of $B/C$ correspond to module automorphisms of $M/N$. In fact, we can choose $M$ and $N$ such that either the extension is split, or $M/N \leq \Phi(G/N)$ (the Frattini subgroup of $M/N$) and $M/N$ is a semisimple module. In the first case, $B/C$ is isomorphic to the full group of module automorphisms, and in the second case we shall show in Proposition 5.1 below that $B = C$. (We are grateful to L.G. Kovács for supplying the proof of this result.)

Finally, to determine $A_N/B$ we need to find the subgroup of $A_M$ consisting of those elements which lift to automorphisms of $G/N$. This subgroup always lies within the subgroup $A'$ of $A_M$ consisting of elements which preserve the isomorphism type of the $G/M$-module $M/N$. In the split case, all elements of $A'$ lift. In the non-split case, we have to work harder and test each individual element of $A'$ for lifting. It is this part of the procedure that can dominate the execution time for the whole calculation, especially when $A_M$ is large. For example, $G/M$ might itself be elementary abelian of fairly small order, but it could have a very large automorphism group.

The computations summarized in the preceding two paragraphs assume that we can perform various calculations on $KG$-modules, where $K$ is a finite field and $G$ is a finite group. These calculations include finding the radical of a module, constructively testing two modules for isomorphism, computing the endomorphism rings of modules, decomposing a module into a direct sum of indecomposable summands, and computing the automorphism group of a module. The radical can be computed from the maximal submodules, which can be found by using the Meataxe methods described in Holt and Rees [1994]. An algorithm for computing endomorphism rings is described in Schneider [1990]; this has been improved recently by C.R. Leedham-Green. Techniques for finding a Fitting element, which can be used to decompose a module into indecomposable summands are also described in Schneider [1990], and Leedham-Green has devised tricks that can be used if such an element cannot be found quickly. A method for isomorphism testing of modules has also been proposed by Leedham-Green. These unpublished algorithms of Leedham-Green have all been implemented in Magma by Allan Steel, and a manuscript describing them is in preparation. An algorithm for module automorphism groups that relies on the other algorithms is described in detail in Smith [1995], and has also been implemented in Magma by Steel. We shall summarize this algorithm in section 5.
In terms of complexity, our algorithm should be comparable to that of Smith, and performance comparisons on straightforward examples with the GAP and Magma implementations of Smith’s algorithm bear this out. However, as indicated above, there are examples of quite small groups (e.g., of order 128) for which it is extremely difficult to compute the automorphism group. Many of the most difficult examples are in fact \( p \)-groups, and in these cases it is preferable to use O’Brien’s \( p \)-group code, which uses more specialized methods.

2.2. The Isomorphism Algorithm

For testing isomorphism between two groups \( G \) and \( H \), the idea roughly is to carry out the automorphism group computations of \( G \) and \( H \) in parallel, maintaining one specific isomorphism \( \xi_i : H/N_i^H \rightarrow G/N_i \) at each layer, where

\[
1 = N_i^H < N_{i-1}^H < \ldots < N_1^H = L^H < H
\]

is the corresponding characteristic series for \( H \) constructed using the same definitions as the series for \( G \). Of course, if the series for \( H \) and \( G \) do not correspond, or if the calculations for \( G/N_i \) and \( H/N_i^H \) fail to correspond at any layer, then we immediately abort the process, because \( G \) and \( H \) are not isomorphic.

For the trivial-Fitting layers, our embeddings \( \rho^H \) and \( \rho^G \) of \( H/N_1^H \) and \( G/N_1 \) into the direct products \( X^H \) and \( X^G \) of wreath products described above are canonical, in the sense that \( H/N_1^H \) and \( G/N_1 \) are isomorphic if and only if \( X^H \) and \( X^G \) are equal and the images of \( \rho^H \) and \( \rho^G \) are conjugate in \( X^G \). If this is the case, then the conjugating element provides the required isomorphism \( \xi_1 \).

During the process of lifting through an elementary abelian layer \( M/N \) of \( G \), we already have an isomorphism \( \xi_M : H/M^H \rightarrow G/M \) and we are looking for an isomorphism \( \xi_N : H/N^{H^H} \rightarrow G/N \). Testing whether such an isomorphism \( \xi_M \) lifts to some \( \xi_N \) is not difficult, and is very similar to testing whether an automorphism in \( A_M \) lifts to one in \( A_N \). (Both of these tests reduce to solving a system of linear equations over a prime field.) The problem is that \( H/N^H \) and \( G/N \) may be isomorphic, but there will not necessarily be an isomorphism \( \xi_N \) that induces our particular \( \xi_M \) on \( H/M^H \). For example, if \( G \) is the nonabelian group of order 21, \( H = G \), \( |N| = 1 \), and \( |M| = 7 \), then a nontrivial automorphism of the group \( G/M \) of order 3 does not lift to an automorphism of \( G/N = G \). In general, \( \xi_N \) induces \( \xi_M \psi_M \) for some \( \psi_M \in \text{Aut}(G/M) \), so we need to consider all such \( \xi_M \psi_M \) as candidates for lifting to \( \xi_N \). However, we clearly do not need to test \( \xi_M \psi_M \) if \( \psi_M \) itself lifts to \( \psi_N \in \text{Aut}(G/N) \).

The convenient way to carry out the search for \( \xi_N \) is to do it during the lifting phase of the computation of \( \text{Aut}(G/N) \) from \( \text{Aut}(G/M) \); for each \( \psi_M \) that does not lift to \( \text{Aut}(G/N) \) during this search, we test \( \xi_M \psi_M \) for lifting to an isomorphism \( \xi_N : H/N^H \rightarrow G/N \). As soon as we find such an isomorphism, then we may of course discontinue this search. On the other hand, if we complete the lifting phase without finding a lifting of any \( \xi_M \psi_M \), then we can conclude
that $H/N^H$ and $G/N$ are not isomorphic, and hence that $H$ and $G$ are not isomorphic.

The bulk of the algorithm descriptions in the remainder of the paper will refer to the computation of $\text{Aut}(G)$. It turns out that isomorphism testing requires surprisingly few new ideas and machinery, and so we shall not give many further details.

2.3. Some Technicalities

In order to carry out all of the above calculations, we shall require presentations of the groups $G/N_i$ (and of $H/N^H_i$ if testing for isomorphism). We shall use these, for example, to decide whether a map $G/N_i \to G/N_j$ defined by generator images is an automorphism. This is most conveniently achieved by a certain type of presentation $\langle X|R \rangle$ of $G$. For ease of notation, we shall allow the set $R$ to consist of a mixture of relators, which are words in the generators, and relations, which are equations between two such words. (Throughout this paper, when we refer to a word in a set $X$ of group elements, we shall mean a product $x_1x_2\ldots x_r$ in which each $x_i \in X \cup X^{-1}$.)

Following Smith [1995], if $M$ is a normal subgroup of a group $G$, then we say that a presentation $\langle X|R \rangle$ of $G$ exhibits $G$ as an extension of $M$ by $G/M$ if

(i) $X = X_1 \cup X_2$, where $X_1 \subseteq G \setminus M$ and $X_2 \subseteq M$;
(ii) $R = R_1 \cup R_2 \cup R_3$ where
   a) $R_1$ is a set of relations of form $w_1 = w_2$, where $w_1$ and $w_2$ are words in $X_1$ and $X_2$, respectively, and the set of all such $w_1$ that occur is a set of defining relators for $G/M$ on $X_1$;
   b) $R_2$ is a set of relations of the form $x^{-1}yx = w$ where $x \in X_1, y \in X_2$ and $w$ is a word in $X_2$, and there is one such relation for each $x \in X_1, y \in X_2$;
   c) $R_3$ is a set of relators $w$, where the words $w$ in $X_2$ form a set of defining relators for $M$ on $X_2$.

Note that the relators in $R_2$ may be used to write any element of $G$ in the form $w_1w_2$ for words $w_1$ and $w_2$ in $X_1$ and $X_2$. We shall call such words normalized.

Extending this idea, if we have a series $N_i$ ($0 \leq i \leq r$) of normal subgroups of a group $G$ with $N_0 = G$, then we say that a presentation of $G$ exhibits the series if, for $2 \leq i \leq r$, on removal of all generators that lie in $N_i$ from the presentation, it exhibits $G/N_i$ as an extension of $N_{i-1}/N_i$ by $G/N_{i-1}$. Thus, the generators in such a presentation lie in disjoint subsets $X_i$ ($1 \leq i \leq r$) with $X_i \subseteq N_{i-1} \setminus N_i$, and any word in $X$ can be rewritten to a normalized word of the form $w_1w_2\ldots w_r$, where $w_i$ is a word in $X_i$. In the soluble case, this is a property of a so-called special PC-presentation of $G$. However, its usefulness is not restricted to soluble groups.

The automorphisms of the quotients $G/N_i$ that we find will be defined by their actions on the generators of such a presentation; that is as automorphisms
of the corresponding quotients of the finitely presented group \( \langle X \mid R \rangle \) that is isomorphic to \( G \). However, in order to compute efficiently within the group of automorphisms, we shall need to be able to calculate the action of composites of several automorphisms and their inverses on arbitrary group elements. This does not seem to be easy for automorphisms of groups defined by a finite presentation, but it can be done for permutation groups as follows.

Let \( G \) be a permutation group with a given generating set \( X \). We shall assume that the reader has some basic familiarity with algorithms for computing in finite permutation groups, including the theory of bases and strong generating sets. See Butler [1991] or Bosma and Cannon [1992], for example. Before attempting to compute with automorphisms of \( G \), we carry out the standard Schreier-Sims algorithm or one of its improved versions to extend the generating set \( X \) of \( G \) to a strong-generating set. Each new strong generator that is introduced during this process is defined as a word in the existing strong generators, and these words are stored. If we are now given the images of the generating set \( X \) under an automorphism \( \alpha \) of \( G \), then these words can be used to compute the image of each of the strong generators under \( \alpha \). Since an arbitrary element \( g \) of \( G \) can easily be expressed as a word in the strong generators, the image \( g^\alpha \) of \( g \) under \( \alpha \) can then be readily computed. In practice, the use of straight-line program datastructures is recommended for these evaluations. Composites of automorphisms can be handled easily using this framework. Inverses of automorphisms \( \alpha \) are currently computed by finding the order \( k \) of \( \alpha \) and then setting \( \alpha^{-1} = \alpha^{k-1} \).

There is a difficulty here in that we have a permutation representation of \( G \), but not of the quotients \( G/N_i \), and we need to be able to compute with automorphisms of \( G/N_i \). In general, if \( G \) is a permutation group of degree \( d \) and \( N \triangleleft G \), then it is not always possible to find a permutation representation of \( G/N \) of degree comparable to \( d \), and we not attempt to do this for \( N = N_i \). Instead, we proceed as follows. For an automorphism \( \overline{\phi} \) of \( G/N_i \), we define a map \( \phi : X \to G_i \) by setting \( x^\phi = g \), for \( x \in X \), where \( g \) is an inverse image of \( (xN_i)^\overline{\phi} \) in \( G \). Of course \( \phi \) is unlikely to define an automorphism or even an endomorphism of \( G \), but we can still carry out the procedure described in the preceding paragraph to compute an element \( g^\phi \) for any \( g \in G \), and we will then have \( g^\phi N_i = (gN_i)^\overline{\phi} \). So we can effectively carry out the required calculations in \( \text{Aut}(G/N_i) \).

We do, however, require a permutation representation of the trivial-Fitting factor group \( G/N_1 = G/L \) of \( G \) in order to find an explicit isomorphism between the socle factors of \( G/L \) and the corresponding simple groups stored in the database. For this purpose we can use a result proved in Luks and Seress [1997] and in Holt [1997] which states that \( G/N \) has a faithful permutation representation of degree at most \( d \) for various particular normal subgroups \( N \) of \( G \), including the largest normal soluble subgroup of \( G \). The proof of this result makes it clear how to calculate the epimorphism from \( G \) to the permutation representation of \( G/L \), and this calculation has been implemented in Magma. (The
existence of this representation can also be deduced immediately from results proved previously by Easdown and Praeger [1988].)

2.4. Identifying an Inner Automorphism

Another problem that we shall encounter is the following: given an inner automorphism $\psi$ of a group $G$, find $x \in G$ such that $g^\psi = g^x$ (where $g^x = x^{-1}gx$) for all $g \in G$. Provided that we have a suitable representation of $G$, such as a permutation or matrix representation, in which centralizers of elements can be computed and the conjugacy problem for elements easily solved, we can proceed as follows.

Let $G = \langle g_1, \ldots, g_r \rangle$. First, find an element $x_1$ in $G$ such that $g_1^\psi = g_1^{x_1}$. Next, find an $x_2$ in $C_G(g_1^\psi)$ such that $g_2^\psi = g_2^{x_1x_2}$. Then find an $x_3$ in $C_G(\langle g_1^\psi, g_2^\psi \rangle)$ such that $g_3^\psi = g_3^{x_1x_2x_3}$. Proceed to find $x_4, \ldots, x_r$ along these lines, and set $x = x_1x_2\cdots x_r$. Then $x$ is the required conjugating element. If any of the conjugacy tests fail, then $\psi$ was not an inner automorphism.

3. The Trivial-Fitting Group Case

In this section, we assume that $G$ is a nontrivial permutation group in which the largest soluble normal subgroup is trivial, and we describe our algorithm for computing $\text{Aut}(G)$ in this special case. We also describe how to test two groups $G$ and $H$ with this property for isomorphism.

3.1. The Theory Behind the Algorithms

Let $G$ be a finite trivial-Fitting group, and let $M$ be the socle $\text{Soc}(G)$ of $G$; that is, the group generated by all minimal normal subgroups of $G$. Since a minimal normal subgroup is a direct product of simple groups (see, for example, Theorem 5.20 of Rotman [1995]), and any two minimal normal subgroups are disjoint, it follows that $M$ is itself a direct product of simple groups, which in this situation all be nonabelian.

Since $M$ has trivial centre, we have $C_G(M) \cap M = 1$, and hence $C_G(M) = 1$, because otherwise $C_G(M)$ would contain a minimal normal subgroup of $G$ disjoint from $M$. Thus the action of $G$ by conjugation on $M$ is faithful, and we can regard $G$ as a subgroup of $\text{Aut}(M)$. Now $G$ has trivial centre, so we can identify $G$ with its inner automorphism group, and regard $G$ as a subgroup of $\text{Aut}(G)$. The socle $M$ is characteristic in $G$ and hence normal in $\text{Aut}(G)$, and we must have $C_{\text{Aut}(G)}(M) = 1$, because otherwise $C_{\text{Aut}(G)}(M)$ would be a normal subgroup of $\text{Aut}(M)$ disjoint from $G$ and would have to centralize $G$. Hence we have $M \subseteq G \subseteq \text{Aut}(G) \subseteq \text{Aut}(M)$, and because $\text{Aut}(G)$ normalizes $G$ and $N_{\text{Aut}(M)}(G)$ acts as a faithful group of automorphisms of $G$, we have $\text{Aut}(G) = N_{\text{Aut}(M)}(G)$.

The idea of the automorphism group algorithm is to find an explicit monomorphism $\rho : G \to \text{Aut}(M)$, and then to compute $\text{Aut}(G)$ as the normalizer of $\text{Im}(\rho)$.
in Aut(M). The actions of specific elements of Aut(G) on G can be computed by using inverse images under ρ. If we are testing two trivial-Fitting groups G and H for isomorphism, then we embed G in Aut(M) and H in Aut(M_H), where M_H = Soc(H). This means G and H are isomorphic if and only if there is an isomorphism from Aut(M) to Aut(M_H) that maps G to H. Such an isomorphism must map M to M_H, and so if M and M_H are not isomorphic, then we can say immediately that G and H are not isomorphic. In fact, when M and M_H are isomorphic, we can organize our implementation so that G and H are both embedded in the same group Aut(M), and then we can test G and H for isomorphism by testing them for conjugacy within Aut(M).

Since the automorphism group of M permutes the direct factors of the socle by conjugation, we see that Aut(M) is a direct product of wreath products Aut(S_i)≀Sym(d_i), where there are d_i factors of the socle isomorphic to the simple group S_i. However, we can gain a small improvement in efficiency in some cases by using a proper subgroup of Aut(M).

Let the simple factors of M be

\[ S_{11}, S_{12}, \ldots, S_{1d_1}, S_{21}, \ldots, S_{2d_2}, \ldots, S_{r1}, \ldots, S_{rd_r}, \]

where S_ij and S_ik are conjugate in G if and only if i = k. Then G embeds in the subgroup of Aut(M) which preserves the G-orbits on the S_ij; that is, G embeds in the direct product of the groups W_i := Aut(S_{i1})≀Sym(d_i), for 1 ≤ i ≤ r.

Let us now rearrange the W_i into equivalence classes

\[ W_{11}, \ldots, W_{1e_1}, W_{21}, \ldots, W_{2e_2}, \ldots, W_{s1}, \ldots, W_{se_s}, \]

where W_{ij} ≅ W_{i'k} if and only if i = i'. Let Y be the direct product of the wreath products W_{ii} ≅ Sym(e_i). Then W ⊆ Y ⊆ Aut(M), and Y contains Aut(G), because Aut(G) must permute the G-orbits on the S_ij. In our algorithm, we construct the embedding of G into Y explicitly, and then compute Aut(G) as the normalizer of G in Y. The referee has pointed out to us that it would be possible to use a still smaller group Y, by using isomorphism between the images of G in the groups W_i as a further refinement when arranging the W_i into equivalence classes, but we have not yet done this in our implementation.

If we are testing two trivial-Fitting groups G and H for isomorphism, then the corresponding groups S_{ij}, W_i and Y for H must be isomorphic to the corresponding subgroups of H, or else G and H could not be isomorphic. In fact, in our implementation, the groups W_i and Y constructed depend only on their isomorphism type. In other words, we embed G and H in the same group Y, and then G and H are isomorphic if and only if they are conjugate in Y.

In the remaining subsections, we describe our implementation of these methods.

3.2. Identifying the Structure of Soc(G)

We first compute the simple factors of the socle of G, and the permutation action of G on these factors induced by conjugation in G. (See Cannon and Holt [1997]
for a description of the algorithms involved here.) As above, let the factors be

\[ S_{11}, S_{12}, \ldots, S_{1d_1}, S_{21}, \ldots, S_{2d_2}, \ldots, S_{r1}, \ldots, S_{rd_r}, \]

where \( S_{ij} \) and \( S_{kl} \) are conjugate in \( G \) if and only if \( i = k \). For each \( i, j \), let \( N_{ij} = N_G(S_{ij})/C_G(S_{ij}) \). Then there is a natural embedding of \( N_{ij} \) into \( \text{Aut}(S_{ij}) \) defined by the conjugation action.

Magma has a library of nonabelian simple groups, which currently includes all such groups of order up to sixteen million, together with a few others, such as alternating groups up to degree 50. This library includes a permutation group \( S \) isomorphic to each simple group and a permutation group \( A \) of minimal degree isomorphic to \( \text{Aut}(S) \), where \( S \leq A \) and the embedding of \( S \) in \( A \) is the standard one where \( S \) is identified with its inner automorphism group. The library also includes finitely presented groups \( S' \) and \( A' \) isomorphic to \( S \) and \( A \), where the generators of \( S' \) and \( A' \) correspond to those of \( S \) and \( A \) under the isomorphism. For up-to-date information on this library, see Bosma and Cannon [2002a].

We need to assume that groups isomorphic to each of the socle factors \( S_{ij} \) of \( G \) are stored in the library; if not, the computation will fail. Our first task is to identify, for each \( i \) with \( 1 \leq i \leq r \), the library group \( S_i \) that is isomorphic to \( S_{i1} \), and also the subgroup \( N_i \) of \( A_i \) that corresponds to the image of \( N_{i1} \) in its embedding into \( \text{Aut}(S_{i1}) \). We will then want to set up the corresponding explicit isomorphisms \( \psi_i : N_{i1} \to A_i \). For a more sophisticated treatment of this problem for the sporadic groups, see Wilson [1996].

In order to carry out these calculations efficiently, we need a permutation representation of \( N_{ij} \) for each \( i \). In this particular situation the action of \( N_G(S_{i1}) \) on the orbits of \( C_G(S_{i1}) \) is very often faithful. If not, then we use some other easily calculated representation of reasonably small degree, such as the conjugation action on Sylow subgroups.

For a fixed \( i \), let \( m = |S_{i1}| \) and \( n = |N_{i1}| \). In most cases, \((m, n)\) suffices for the identification of \( N_i \) and \( S_i \). In ambiguous cases, such as \((m, n) = (360, 720)\) or \((20160, 20160)\), we use the sum of the orders of representatives of the conjugacy classes as a convenient and easily calculated invariant to distinguish between them.

We can then extract the relevant information about \( S_i \) and \( N_i \) from the library. Recall that finitely presented groups \( S'_i \) and \( A'_i \) that are isomorphic to \( S_i \) and \( A_i \) are stored. The group \( S'_i \) is defined on two generators \( x'_1 \) and \( x'_2 \), chosen such that the conjugacy class of \( x'_1 \) in \( S'_i \) is uniquely determined by the order of \( x'_1 \) and the length of the class. We conjecture that such classes exist in all finite simple groups, although we do not know how to prove this. Subject to this restriction, \( x'_1 \) and \( x'_2 \) are chosen such that the expected time to construct the isomorphism between \( S \) and \( S' \) is as small as possible; the construction of this isomorphism is described below. Generators \( z'_1, z'_2, \ldots, A'_i \) modulo \( S'_i \) are also stored in the library, together with the conjugation action of each such \( z'_j \) on \( S'_i \). This action is stored as a list of the images of the generators \( x'_1, x'_2 \) given as words in \( x'_1, x'_2 \).
A representative $N'_i/S'_i$ of each conjugacy class of subgroups of $A'_i/S'_i$ has been chosen and stored. For each such $N'_i$, generators $x'_3, x'_4, \ldots$ of $N'_i$ modulo $S'_i$ are stored as words in the $z'_j$.

For example, suppose that $S'_i = A_6 \cong \text{PSL}(2,9)$. Let us omit the ‘prime’ symbols and the subscript $i$ while describing this example. The stored presentation of $S$ is

$$\langle x_1, x_2 \mid x_1^3, x_2^3, (x_1x_2)^4, (x_1x_2^{-1})^5, [x, y]^5 \rangle.$$ 

There are two generating outer automorphisms, $z_1$ and $z_2$, which correspond, respectively, to elements in the subgroups $S_6 \cong \text{PΣL}(2,9)$ and $\text{PGL}(2,9)$ of $\text{Aut}(S) \cong \text{PTL}(2,9)$. We chose $z_1$ to map $x_1$ to $x_1$ and $x_2$ to $x_1^{-1}x_2x_1x_2^{-1}x_1^{-1}x_2^{-1}x_1$, and $z_2$ to map $x_1$ to $x_2$ and $x_2$ to $x_1$. There are five subgroups $N$:

1. $N = S = \langle x_1, x_2 \rangle$;
2. $N = \langle x_1, x_2, x_3 \rangle \cong S_6$, with $x_3 = z_1$;
3. $N = \langle x_1, x_2, x_3 \rangle \cong \text{PGL}(2,9)$, with $x_3 = z_2$;
4. $N = \langle x_1, x_2, x_3 \rangle \cong M_{10}$, with $x_3 = z_1z_2$;
5. $N = A = \langle x_1, x_2, x_3, x_4 \rangle$, with $x_3 = z_1, x_4 = z_2$.

Notice that the cases (ii), (iii) and (iv) in this example cannot be distinguished by using their orders alone, but the sum of the orders of representatives of their conjugacy classes are respectively 38, 58 and 35, and this invariant is used to identify the group $N$ in these cases.

Returning now to the general description, there is an additional complication when $N'_i$ is not normal in $A'_i$. In this situation, coset representatives of $N_{A'_i}(N'_i)$ in $A'_i$ are stored in the library again as words in the $z'_j$. We shall see below why this is necessary.

So, having now extracted the above information from the library, the next step is to define an explicit isomorphism $\phi_i : N'_i \to N_{i1}$. To do this, we first define the restriction $\phi_i : S'_i \to S_{i1}$. We need to find the images $x_1, x_2 \in S_{i1}$ of $x'_1, x'_2 \in S'_i$ under $\phi_i$. We do this by choosing random elements of $S_{i1}$. Because of our choice of $x'_1$ as being in a conjugacy class that is unique for its order and length, finding and identifying an appropriate $x_1$ is straightforward. Having found $x_1$, we search for an element $x_2 \in S_{i1}$ such that $x_1$ and $x_2$ satisfy the defining relations of $S'_i$.

If $N_i > S_{i1}$, then we need to extend $\phi_i$ to an isomorphism from $N'_i$ to $N_{i1}$, by finding images $x_3, x_4, \ldots \in N_{i1}$ of the stored generators $x'_3, x'_4, \ldots$ of $N'_i$. From the stored action of the $z'_j$ on $S'_i$, we can use $\phi_i$ to calculate the corresponding automorphisms of $S_{i1}$, and then calculate the automorphisms of $S_{i1}$ corresponding to $x'_3, x'_4, \ldots$. We then have to find corresponding elements $x_3, x_4, \ldots$ of $N_{G}(S_{i1})$ that induce these automorphisms by conjugation, and we described how to do that in subsection 2.4.

However, the complication when $N'_i$ is not normal in $A'_i$ arises at this point. Our group $N_{G}(S_{i1})$ may well correspond to a conjugate of $N'_i$ in $A'_i$ rather than to $N'_i$ itself, in which case we will not be able to find $x';$ in fact one of the two conjugacy tests will fail. In this case, we can make $N_{G}(S_{i1})$ correspond to
$N'_i$ by changing $\phi : S'_i \to S_{ij}$. For each coset representative $z$ of $N_{A'_i}(N'_i)$ in $N'_i$ in turn, we replace $x_1$ and $x_2$ by their images under the automorphism of $S_{ij}$ corresponding to $z$ (and we must always use the original $x_1$ and $x_2$ in this calculation, not the most recent replacements!). For exactly one of these coset representatives $z$, we will find the required elements $x_3, x_4, \ldots$ of $N_G(S_{11})$, and then we can stop.

Since the generators of the stored permutation group $A_i$ have been chosen to correspond to those of $A'_i$, we can now use $\phi_i$ to construct the desired embedding $\psi_i : N_{i1} \to A_i$ with image $N_i$. Summing up, for each orbit of the group $G$ on the socle factors, we have constructed a homomorphism $\psi_i : N_G(S_{11})/C_G(S_{11}) \to A_i$.

The next step is to use these maps to construct homomorphisms $\rho_i : G \to W_i$, again one for each orbit on the socle factors, where $W_i$ is the wreath product $A_i \wr \text{Sym}(d_i)$. This construction has been described a number of times in the literature. See, for example, Gross and Kovács [1984]. For each $j$ with $1 \leq j \leq d_i$, $\rho_i$ maps $S_{ij}$ isomorphically onto one of the socle factors of the base group of $W_i$, where $\rho_i$ defines an equivalence between the permutation actions by conjugation of $G$ on the $S_{ij}$ and of $G^\rho$ on the socle factors of the base group of $W_i$.

These homomorphisms $\rho_i$ together define a map $\rho = (\rho_1, \ldots, \rho_r)$ from $G$ to the direct product $X$ of the $W_i$ and $\rho$ is clearly a monomorphism, because the $S_{ij}$ do not lie in the kernel of $\rho_i$, so none of the socle factors lie in the kernel of $\rho$. Let $V$ be the image of $\rho$; so $G \cong V$.

Finally, later in the automorphism group computations, it will be convenient to have a fixed generating set for $G$ and a method for writing elements in $G$ into a normal form in those generators. We know that $M = \text{Soc}(G)$ is the direct product of the simple groups $S_{ij}$. Our generating set for $G$ will contain generators of each of the groups $S_{ij}$ together with a small number of extra generators outside of $M$. For each coset of $M$ in $G$, we choose a fixed word in these extra generators which lies in that coset. (Here, we are making use of the fact that $G/M$ is quite small.) Then our normal form for elements of $G$ will be of the type $uv_1 \ldots v_r$ where $u$ is our chosen word for the appropriate coset of $M$ in $G$, and each $v_k$ is a word in the generators of one of the socle factors $S_{ij}$ of $M$. To obtain a normal form for the words $v_k$, we either use a shortest word in the generating set of size two that was obtained from the library, and which is easily calculated when $S_{ij}$ is small or, for large $S_{ij}$, we use a word in a strong generating set for a permutation representation of $S_{ij}$. Although there is a potential problem involved here in that such words in strong generating sets can sometimes be very long, we have not yet found it necessary to use any specialized methods for avoiding this problem.

### 3.3. Computing $\text{Aut}(G)$

Now the automorphism group $\text{Aut}(G)$ of $G$ certainly permutes the socle factors of $G$, and indeed it permutes the orbits of $G$ on the socle factors. If the $i$-th and the $j$-th of these orbits are fused under $\text{Aut}(G)$, then we must have $d_i = d_j$, and $S_{i1}$ and $S_{j1}$ must be isomorphic. But then, because the stored representations
So we rearrange the wreath products $W_i$ into equivalence classes

\[ W_{11}, \ldots, W_{1e_1}, W_{21}, \ldots, W_{2e_2}, \ldots, W_{s1}, \ldots, W_{se_s} \]

where $W_{ij} = W_{ik}$ if and only if $i = i'$, and embed the direct product $X$ of the $W_{ij}$ in the natural way in the group $Y$, which is defined to be the direct product of the wreath products $W_{ii} \wr \text{Sym}(e_i)$ for $1 \leq i \leq s$. As explained in subsection 3.1, $\text{Aut}(G)$ is isomorphic to the normalizer of $V \cong G$ in $Y$.

So we use this normalizer computation to compute $\text{Aut}(G)$. We need to compute the action of generators of $\text{Aut}(G)$ on generators of $G$, and also a presentation of $\text{Aut}(G)$ with the required properties. Despite the apparent complexity of the above constructions, the index of $M \cong \text{Soc}(V)$ in $V$ and also in $\text{N}_Y(V)$ is typically quite small, and so getting a presentation of $\text{N}_Y(V)/\text{Soc}(V)$ is not difficult. The required computations can then be carried out provided that we can get a presentation of $M$, and express elements of $M$ as words in a suitable generating set of $M$. Since $M$ is a direct product of simple groups, these problems reduce easily to the case when $M$ is itself simple, and in fact the most awkward situation tends to be when $M$ is a large simple group. In that case, it is expedient to use a presentation of $M$ on a strong generating set. This tends to have a large number of generators and relations, but it has the advantages that it is relatively easy to calculate, the relations are quite short, and it easy to express an element in a permutation group as a word in the strong generators.

### 3.4. Testing $G$ and $H$ for Isomorphism

Suppose now that we want to test two groups $H$ and $G$ with trivial Fitting subgroups for isomorphism. We carry out the computations described above for both $G$ and $H$ and, in particular, we calculate the groups $X^H$ and $Y^H$ for $H$ that correspond to $X$ and $Y$ for $G$ and the map $\rho^H : H \to X^H$ corresponding to $\rho$. We may assume that $X = X^H$ and $Y = Y^H$, because otherwise $H$ and $G$ would not be isomorphic, and similarly we may assume that $G$ and $H$ have the same orders.

As we saw in subsection 3.1, $G$ and $H$ are isomorphic if and only if their images under $\rho$ and $\rho^H$ are conjugate in $Y$. If they are conjugate, then we can use a conjugating element and the inverses of the maps $\rho$ and $\rho^H$ to construct an explicit isomorphism from $H$ to $G$.

### 4. Finding the Characteristic Series

In this section we describe how to find the series of characteristic subgroups

\[ 1 = N_r < N_{r-1} < \ldots < N_1 = L \leq G \]
of $G$, such that each $N_i/N_{i+1}$ is elementary abelian and $G/L$ is a trivial-Fitting group. We also describe how to find a presentation which exhibits this series.

Our algorithm for finding the series is quite simple and uses standard group-theoretical algorithms. (See Bosma and Cannon [1992] for details and references.) We start by finding the soluble radical $L$ which, in the permutation group context, involves reductions to deal with the intransitive and imprimitive cases; see Unger [2002]. An algorithm for computing $L$ is also described in Luks and Seress [1997]. We then compute the derived series of $L$, each layer of which is divided up into characteristic elementary abelian sections by taking subgroups generated by $p$-th powers, where the primes $p$ dividing the layer are taken in increasing order. In order for this process to work correctly for isomorphism testing it is of course essential that the series be computed in a well-defined manner, so that it is guaranteed to give results that correspond properly when applied to isomorphic groups.

Experience shows that, at least in difficult examples, it is more efficient in automorphism group calculations to split large layers up into smaller ones if possible, even though this means increasing the length of the series. We can attempt to refine a factor $M/N$, by regarding it as a $G/M$-module with module action induced by conjugation in $G$, and looking for ‘characteristic’ submodules, such as the radical. We can also try intersecting $M$ with other characteristic subgroups of $G$, such as terms in the central series. Again, when isomorphism testing we must make sure that we carry out the same operations in $G$ and in $H$. There is also one situation, which we shall encounter in subsection 5.1 below, where we need to refine the series in the middle of the automorphism group computation.

We need to find a generating set of $G$ that exhibits this series. We do this by finding appropriate generators starting from the top down, beginning with $G/N_1 = G/L$. If this group is nontrivial then, as mentioned in subsection 2.3, we can find a faithful permutation representation of $G/L$ of degree at most that of $G$, and we use this representation and the method described in section 3 to compute $Aut(G/L)$.

The full generating set $X$ is the disjoint union of $X_1 = \{y_1, y_2, \ldots, y_d\}$, where the $y_i$ map onto the generators $x_i$ of $U = G/L$ and, for $2 \leq i \leq r$, $X_i = \{m_{i1}, m_{i2}, \ldots, m_{id}\}$, where $N_{i-1}/N_i$ is elementary abelian of order $p_i^{d_i}$ for some prime $p_i$. Such generators are easily found by choosing random elements. We can then define an epimorphism $\phi_i$ from $N_{i-1}$ to an elementary abelian group $M_i$ of order $p_i^{d_i}$ with $\ker(\phi_i) = N_i$. By evaluating images and inverse images under $\phi_i$, it is then straightforward to compute $d_i \times d_i$ matrices over $GF(p_i)$ which represent the conjugation action of the generators of $G$ on the layer $N_{i-1}/N_i$, and hence to set $M_i$ up as a $G/N_{i-1}$-module over $GF(p_i)$.

It is convenient to have a normal form in these generators for elements of $G$. This will have the structure of a normalized word $w_1 w_2 \ldots w_r$, where $w$ is a word in $X_i$, as discussed above in subsection 2.3. A suitable normal form for $w_1$ was described at the end of subsection 3.2. For $i > 1$, there is an obvious normal
form for $w_i$, which is $m_{i_1}^{\alpha_1}m_{i_2}^{\alpha_2}\ldots m_{i_d_i}^{\alpha_{d_i}}$, where $0 \leq \alpha_j < p_i$. It is not difficult to compute the normal form word for a permutation $g \in G$. We first map $g$ onto $G/L$ and use the methods of subsection 3.2 to find $w_1$. We then replace $g$ by $g_1^{-1}g$, where $g_1$ is a well-defined inverse image of $w_1$ in $G$. Next, we use $g_2^{-1}g$ to compute $w_2$, and replace $g$ by $g_2^{-1}g$, where $g_2$ is a well-defined inverse image of $w_2$ under $\phi_2$. Proceeding in this fashion, we can find each $w_i$ in turn.

Given that we can compute this normal form for elements of $G$, it is straightforward to determine the relators $R$ in the required presentation $\langle X \mid R \rangle$ of $G$ which exhibits the series. We will also use the normal form when specifying images of group generators under elements of Aut($G$).

5. Lifting Automorphisms to the Next Layer

In this section, we assume that $G$ is a group with normal subgroups $N$ and $M$, such that $N \leq M$ and $M/N$ is an elementary abelian $p$-group of order $p^d$ for some prime $p$ and some integer $d > 0$. We describe how to compute Aut($G/N$) under the assumption that Aut($G/M$) has already been calculated. Since everything under discussion concerns the quotient group $G/N$ rather than $G$, we shall simplify notation and assume throughout the remainder of the section that $N = 1$. In particular $G/N$ and $M/N$ will henceforth be written simply as $G$ and $M$, respectively. Occasionally we shall need to remind the reader that calculations are in fact being carried out modulo a normal subgroup $N$.

The methods that we use are based strongly on those described for soluble groups by Smith [1995], but we have introduced a number of novel features. In the case when the extension $G$ of $M$ by $G/M$ does not split, we refine the series of normal subgroups of $G$ if necessary to get $M \subseteq \Phi(G)$. This has the advantage that we do not get any automorphisms centralizing $G/M$ but not $M$. Our method of testing whether a particular automorphism of $G/M$ lifts to one of $G$ is essentially the same as in Smith [1995], but we use a completely different method for searching through Aut($G/M$). We represent Aut($G/M$) as a permutation group, and use a backtrack search through its elements. This method can be time-consuming but it uses very little space, whereas the Orbit-Stabilizer method described in Smith [1995] will often fail as a result of running out of memory. Finally, we compute a group presentation of Aut($G$), whereas only generators are found in Smith [1995].

We assume that we have a presentation of $G$ which exhibits this extension of $M$ by $G/M$. Let the generating set for this presentation be $Y = Y_1 \cup Y_2$ where $Y_1$ generates $G$ modulo $M$ and $Y_2$ generates $M$. If we are testing two groups for isomorphism, then we also have a presentation for $H$, which exhibits it as an extension of $M^H$ by $H/M^H$.

We assume that the group $A_M$ of automorphisms of $G/M$ has already been computed during the previous layer calculations (except when $G/M = 1$). In the isomorphism testing situation, we will have computed $A_M$ for $G$, together with one specific isomorphism $\xi_M : H/M^H \to G/M$. We need to be a little more
precise about the form in which $A_M$ is specified. The reader can check from section 3 that we have conformed to this specification in our calculation of $A_L$ for the trivial-Fitting top factor $G/L$.

Let $I_M$ denote the group of inner automorphisms of $G/M$. Then $I_M$ is isomorphic to $(G/M)/Z(G/M)$, and generators of the centre $Z(G/M)$ of $G/M$ as words in $Y_1$ will have been calculated. (Note that the top factor $G/L$ has trivial centre.) Let $\gamma_M : G/M \to I_M$ be the natural map with kernel $Z(G/M)$. Then a set $O_M$ of automorphisms of $G/M$ which generate $A_M$ modulo $I_M$, together with a presentation of $A_M$ on $Y_1^{\gamma_M} \cup O_M$ which exhibits $A_M$ as an extension of $I_M$ by $A_M/I_M$ will have been computed.

Let $A = \text{Aut}(G)$, let $I$ be the group of inner automorphisms of $G$, and let $\gamma : G \to I$ be the natural map with kernel $Z = Z(G)$. As explained in subsection 2.1, $A$ has normal subgroups $C \leq B$, where $B$ consists of those automorphisms which induce the identity on $G/M$, and $C$ consists of those which induce the identity on both $G/M$ and $M$. We calculate $A/I$ in three stages, which consist respectively of the computations of $CI/I$, $BI/CI$ and $A/BI$. These will be described in detail in subsections 5.1, 5.2 and 5.3 below. In fact we shall find sets of outer automorphisms $O_C, O_B$ and $O_A$ of $G$ lying in $C, B$ and $A$ respectively, generating $CI$ modulo $I$, $BI$ modulo $CI$ and $A$ modulo $BI$, respectively. Generators of $Z = \text{Ker}(\gamma)$ as words in $Y$ will be found during the first stage. Generators of $C \cap I$ and $B \cap I$ will also be found as words in $Y^\gamma$; it is easy to see that $C \cap I = C_{Z_M}(M)^\gamma$ and $B \cap I = Z_M^\gamma$, where $Z_M$ is the inverse image of $Z(G/M)$ in $G$.

Finally, we calculate a presentation of $A$ on the generating set $Y^\gamma \cup O_C \cup O_B \cup O_A$. This presentation will exhibit the series

$$1 \leq I \leq CI \leq BI \leq A$$

of $A$. It may be used in the next stage in order to find a suitable permutation representation of $A$ if a backtrack search through its elements should be necessary. This calculation will be described in subsection 5.4. The complete situation is illustrated in Fig. 1.

The second and third stages of the computation of $A$ are substantially different depending on whether the extension $G$ of $M$ by $G/M$ is split or not. It is convenient to decide whether or not this is the case during the first stage, and if the extension splits, then we change generators so as to make $Y_1$ generate a complement of $M$ in $G$.

5.1. Automorphisms Centralizing both $M$ and $G/M$

An automorphism $\phi$ of $G$ which induces the identity on both $G/M$ and $M$ has the form $x^\phi = x(xM)^\tau$, where $\tau : G/M \to M$ is a map. It is easy to check that $(xy)^\phi = x^\phi y^\phi$ for all $x, y \in G$ if and only if $(uv)^\tau = (u^\tau)^v v^\tau$ for all $u, v \in G/M$. Hence the calculation of the group $C$ reduces to calculating maps $\tau : G/M \to M$ that satisfy this identity. Such maps $\tau$ are called derivations or
crossed homomorphisms. The set of all derivations forms an elementary abelian $p$-group isomorphic to $C$ under pointwise multiplication of maps. Calculating it is not difficult and reduces to the solution of a system of linear equations over the field $K = \text{GF}(p)$.

By solving a related system of equations, we can determine whether the extension $G$ of $M$ by $G/M$ splits and, if so, find a map $\rho : G/M \to M$ such that the set $\{x(xM)^{\rho} \mid x \in Y_1\}$ generates a complement of $M$ in $G/M$. In the split case, we replace our generating set of $G$ if necessary, in order to make the images in $G$ of the generators in $Y_1$ generate such a complement.

In the nonsplit case, for reasons which will be explained in subsection 5.2, we would like $M$ to be semisimple as a $G/M$-module, and to have $M$ contained in the Frattini subgroup $\Phi(G)$ of $G$. If this is not the case already, then we can achieve it by refining the series of characteristic subgroups of $G$. First we introduce the inverse image of the radical of the module $M$ into the series to ensure that the new $M$ is semisimple, and then we introduce the inverse image of $\Phi(G)$ into the series. We have now replaced $N$ (which we are assuming to be
trivial for the sake of simplifying notation) by a larger characteristic subgroup \( N' \) lying strictly between \( N \) and \( M \), and we restart the lifting calculations using \( N' \) in place of \( N \).

Full details of the calculations involving derivations can be found in section 4 of Cannon et al. [2001], so we shall not repeat them here. The solution set of the system of linear equations, which corresponds to \( C \), is described as the nullspace of a certain \( dr \times ds \) matrix \( E \), where \( |M| = p^d \) and \( r \) and \( s \) are respectively the numbers of generators and relators in the presentation of \( G/M \). Given an arbitrary automorphism in \( C \), it is straightforward to calculate the corresponding derivation, which can then be expressed as an element of this nullspace, and then expressed as a linear sum of the generators of this nullspace. In other words, we can easily express a given automorphism in \( C \) as a word in a set of generators of \( C \).

In fact, we are trying to compute \( C I/I \cong C/(C \cap I) \), so we need to find a set of generators \( O_C \) of \( C \) modulo \( (C \cap I) \). Now the inner automorphism \( g^\gamma \) induced by \( g \in G \) lies in \( C \) if and only if \( g \) lies in the group \( D := C_{Z_M}(M) \) (see Fig. 1). Since we know generators of \( Z_M \) from the previous layer calculations, we can compute \( D \). (More precisely, we calculate \( D \) as the centralizer of \( M \) in \( Z_M \). At this stage, the group \( G \), and hence also \( Z_M \), is a quotient of a permutation group, but we can use an algorithm described in Kantor and Luks [1990] for the computation of section centralizers in permutation groups, which avoids the explicit formation of quotient groups of permutation groups.) The final generating set of \( C \) that we compute is the union of \( I_C \) and \( O_C \), where \( I_C \) generates \( C \cap I \). For each \( \phi \) in \( I_C \), we compute and store a word \( w \) in \( Y \) such that \( w^\gamma = \phi \). We shall need these words in the calculation of the presentation of \( A \) to be described in subsection 5.4 below.

As mentioned in the preceding paragraph, we can express elements of \( C \) as words in a generating set of \( C \), and so we can calculate the images of the generators of \( D \) under \( \gamma \) and, since \( C \) is elementary abelian, we can compute generators for the kernel of \( \gamma \), which is precisely \( Z \).

If we are testing two groups \( H \) and \( G \) for isomorphism, then we carry out the corresponding calculations in \( H \). In particular, we calculate the corresponding matrix \( E^H \) for the system of linear equations, because we shall need this later.

Of course, if the results of the calculations do not correspond properly with those of \( G \), then \( H \) and \( G \) are not isomorphic, and we can abort the whole procedure.

### 5.2. Automorphisms Centralizing \( G/M \) but not \( M \)

We denote the group of automorphisms of \( G \) that induce the identity on \( G/M \) by \( B \). If \( \phi \) is such an automorphism, \( m \in M \) and \( g \in G \), then using the commutativity of \( M \), we have \((m^\phi)^g = (m^\phi^g)\phi = (m^\phi)^g\). In other words, the restriction of \( \phi \) to \( M \) is an automorphism of the group \( M \) regarded as a \( KG \)-module, where \( K = GF(p) \).

An algorithm for computing the group \( \text{Aut}_{KG}(M) \) of module automorphisms
of $M$ is described and proved correct in section 4.6 of Smith [1995], and has been implemented in Magma by Allan Steel. The restriction of automorphisms in $B$ to $M$ induces a monomorphism $B/C \rightarrow \text{Aut}_{KG}(M)$.

We shall now briefly summarize this algorithm. We assume that we are able to compute endomorphism rings of $KG$-modules, test modules for isomorphism, and decompose modules into indecomposable summands. We first decompose $M$ into direct summands $M_1, \ldots, M_r$, where each individual $M_i$ is a direct sum of $r_i$ isomorphic indecomposable summands $M_{ij}$, and distinct $M_{i_1}, M_{j_1}$ are not isomorphic. For each $i$, we first compute $E_i := \text{End}_{KG}(M_{i_1})$ and its radical $\text{Rad}(E_i)$. It turns out that $\text{Aut}_{KG}(M_{i_1})$ is an extension of a $p$-group of order $|\text{Rad}(E_i)|$ by the multiplicative group of the finite field $F_i := E_i/\text{Rad}(E_i)$, and this structure enables generators to be written down. Since $\text{End}_{KG}(M_i)$ is isomorphic to the algebra of all $r_i \times r_i$ matrices over $E_i$, generators of $\text{Aut}_{KG}(M_i)$ can be constructed using the generators of $\text{Aut}_{KG}(M_{i_1})$, together with two generators of $\text{GL}(r_i, F_i)$. Finally, for each distinct $i, j$, we compute $\text{Hom}_{KG}(M_{i_1}, M_{j_1})$. Elements $\phi$ in these groups give rise to automorphisms of $M$ that induce the identity on each $M_i$, but map elements $x \in M_{i_1}$ to $x + y$ where $y \in M_{j_1}$ is the image of $x$ under $\phi$. It can be shown that $\text{Aut}_{KG}(M)$ is generated by these elements together with generators of each individual $\text{Aut}_{KG}(M_i)$. See Smith [1995] for further details and proofs.

Let us first consider the split case. Then, for each $\phi \in \text{Aut}_{KG}(M)$, we can define an automorphism of $G$ which induces $\phi$ on $M$ and the identity on a given complement of $M$ in $G$. It follows that the map $B/C \rightarrow \text{Aut}_{KG}(M)$ is an isomorphism. Since our generators in $Y_1$ generate such a complement (modulo $N$), and the generators in $Y_2$ are the module generators, for a given $\phi$, we can immediately define the action of the corresponding automorphism of $G$ on the generators of $G$. We take $O_B$ to be a set of generators of $B$ modulo $C$ which is in one-one correspondence with a set of generators of $\text{Aut}_{KG}(M)$ in its representation as a matrix group. If the matrix group is large, then we use a strong generating set, because we will need to be able to write its elements as words in the generators.

As in the calculation of $C$ described in subsection 5.1, we also need to find the subgroup $B \cap I$, but this is just $Z^*_M$. We can now find a set $I_B$ that generates $B \cap I$ modulo $C \cap I$ and, as in the corresponding situation in subsection 5.1, for each $\phi \in I_B$, we store a word $w$ in $Y$ with $w^\gamma = \phi$. For each such $\phi$, we also store a word $w'$ in the generating set $O_B$ with the property that $\phi$ and $w'$ induce the same action on $M$. Finally we store the matrix actions of these elements $\phi$ on $M$.

Notice that, unlike in subsection 5.1, we are taking $O_B$ to be a full generating set of $B$ modulo $C$ rather than of $B$ modulo $(B \cap CI)$. This is because we need to be able to identify elements of $B/C$ as words in $O_B$ from the matrix group representation of $B/C$, and so we need generators which correspond to matrix group generators.

If $B/C$ is fairly small, then we can compute a presentation of it and express elements of $B$ (modulo $C$) as words in $O_B$, by working in the regular permutation
representation of $B/C$. If $B$ is large however, then we use a strong generating set $S$, say, for $C/B$ in its representation as a matrix group, and we can then find a presentation of this group on $S$. (The method of finding a presentation of a matrix group on a strong generating set is similar to that for a permutation group; see, for example, sections 5 and 7 of Neuber
er [1982].) In either case, we can get a presentation of $B/I\cong B/(B\cap CI)$ by adding the generators in $I_B$, for which we have expressions as words in $O_B$, as extra relators.

Turning now to the non-split case, we explained in subsection 5.1 that we are assuming that $M$ is a semisimple module, and that $M \subset \Phi(G)$. As we shall now prove, these conditions imply that $B = C$. However, we still need to calculate the matrix group $\text{Aut}_{KG}(M)$, because we shall need this in the calculation of $O_A$ to be described in subsection 5.3. If we are testing two groups $H$ and $G$ for isomorphism, then we also compute the corresponding module automorphism group $\text{Aut}_{KH}(M^H)$ for $H$, and if its order is not the same as that of $\text{Aut}_{KG}(M)$ then $H$ and $G$ are not isomorphic, and we abort the process.

The fact that $B = C$ follows from the following result, the proof of which is due to Kovács (personal communication).

**Proposition 5.1:** Let $M$ be a normal elementary abelian $p$-subgroup of a group $G$ with $M \leq \Phi(G)$ such that $M$ is semisimple when considered as a $G/M$-module under the conjugation action of $G$. Then any automorphism $\psi$ of $G$ that fixes $M$ and induces the identity map on $G/M$ also induces the identity map on $M$.

**Proof:** Let $Q = G/M$ and $A = \langle \psi \rangle$. Then, since $\psi$ induces the identity map on $Q$, we can consider $M$ as an $(A \times Q)$-module. Clearly we may assume that $A$ has prime power order.

First suppose that $A$ is a $q$-group for some prime $q \neq p$. Then $A$ is a Sylow subgroup of $AM$ in the semidirect product $AG$ and, by the Frattini argument, the normalizer $N$ of $A$ in $AG$ satisfies $AG = AMN = MN$. Hence, since $M \leq G$, we get $G = M(N \cap G)$ and then $M \leq \Phi(G)$ implies $G = N \cap G$. Thus $G \leq N$ and so $G$ normalizes $A$ and hence $A$ centralizes the whole of $G$.

Otherwise $A$ is a $p$-group, in which case $A$ centralizes each irreducible composition factor of $M$. Since $M$ is semisimple as a $Q$-module, by factoring out a minimal $(A \times Q)$-submodule of $M$, we easily reduce to the case when $M$ has just two $Q$-composition factors, and $A \times Q$ acts uniserially on $M$. Let $N$ be an irreducible $Q$-submodule of $M$ that is not fixed by $A$. Then $M = N \times N^\psi$. For each $g \in G$, we know that $g^\psi$ is congruent to $g$ modulo $M$; that is, there exist unique $x, y$ in $N$ such that $g^\psi = gxy^\psi$. It is now straightforward to calculate that the map defined by $g \mapsto gx$ is an endomorphism of $G$ whose kernel is $N$ and whose image avoids and therefore complements $N$. This contradicts the assumption that $M \leq \Phi(G)$, and the contradiction completes the proof.
5.3. Automorphisms which do not Centralize $G/M$

Now we want to find a set $O_A$ of outer automorphisms of $G$ which, together with inner automorphisms, generates $A$ modulo $B$. The natural map $A \to A_M$, mapping an element of $A$ to its induced action on $G/M$ has kernel $B$, so we get an induced monomorphism $\rho : A/B \to A_M$.

The first problem is to decide whether a given automorphism $\overline{\psi} \in A_M$ is in the image of $\rho$ and, if so, to find $\psi \in A$ with $\psi^\rho = \overline{\psi}$. If there is such a $\psi$, then we have $(m^g)^\psi = (m^\psi)^g$ for all $m \in M$ and $g \in G$. However, since $M$ is abelian, the conjugation action of $G$ on $M$ can be regarded as a module action of $G/M$, and we can write this equation as $(m^g)^\psi = (m^\psi)^g\overline{m}$ where $\overline{g} = gM$. In other words, the restriction $\psi_M$ of $\psi$ to $M$ is a $G/M$-module isomorphism between the modules $M$ and $M\overline{g}$, where the action of $\overline{g}$ on $m$ in the latter module is defined to be $m\overline{g}$. We now test whether $M$ and $M\overline{g}$ are isomorphic. (See the remarks at the end of subsection 2.1 concerning algorithms for $K\overline{G}$-modules.) If not, then $\overline{\psi}$ is not in the image of $\rho$.

If the modules are isomorphic, then we can find a specific module isomorphism $\phi_0 : M \to M\overline{g}$. If $\psi^\rho = \overline{\psi}$, then $\psi_M = \phi_0\phi$, for some $\phi$, where $\phi : M \to M$ is a module automorphism of $M$ with the normal (conjugation) action of $G/M$. (To see this, note that $(m^g)^\psi = (m^\psi)^g\overline{m}$ and $(m^g)^\phi_0 = (m^{\phi_0})^{\overline{g}}$ and so if the group automorphism $\phi : M \to M$ is defined by $\psi_M = \phi_0\phi$, then $\phi$ satisfies $(m^g)^\phi = (m^\phi)^g$ for all $m \in M$ and $g \in G$.) Since we calculated this module automorphism group earlier, as described in section 5.2, we now know all possible candidates for $\psi_M$.

The case when the extension of $M$ by $G/M$ is split is straightforward. We can find a suitable $\psi$ with $\psi^\rho = \overline{\psi}$ which normalizes the complement of $M$ in $G/M$ generated by $Y_1$, by letting $\psi$ act as $\phi_0$ on $M$ and as $\overline{\psi}$ on the complement, which is naturally isomorphic to $G/M$. It is routine to check that $\psi$ is an automorphism of $G/M$.

The nonsplit case is more difficult, and it is not necessarily the case that $\overline{\psi} \in \text{Im}(\rho)$ even if $M$ and $M\overline{g}$ are isomorphic as $G/M$-modules. We know from Proposition 5.1 that if an inverse image $\psi$ of $\overline{\psi}$ does exist, then $\psi_M$ is uniquely determined by $\overline{\psi}$. We do not currently know a general method of calculating $\psi_M$ explicitly from $\overline{\psi}$. However, this can usually be done in the particular situation where $M$ lies in the Frattini subgroup of $C_G(M)$, which is useful, because in practice many of the longest searches occur in just this situation. The idea is to try to find relators of $C_G(M)$ in which all generators have exponent sum divisible by the prime $p$ dividing $|M|$. Such relators evaluate unambiguously (i.e. independently of the coset representatives chosen) to elements of $M$. We must find enough such relators such that their values in $M$ generate $M$ as an abelian group, and then we can compute $\psi_M$ simply by applying $\overline{\psi}$ to these relators and computing the result again in $M$. In other situations, or if we fail to find sufficient relators, then we have to try each module automorphism $\phi$ in turn and test whether there is an inverse image $\psi$ with $\psi_M = \phi_0\phi$. The test itself, which
we describe in the next paragraph, is not too expensive; the problem arises when
the module automorphism group is large, and we have a lot of candidates for
\(\psi_M\) which we need to test individually.

Suppose now that we are given \(\psi \in \text{Aut}(G/M)\), and we are looking for
\(\psi \in \text{Aut}(G)\) with \(\psi^\rho = \psi\) such that \(\psi_M\) is a specified map. We still need to
find the action of \(\psi\) on the generators \(Y_1\). We know that, for each generator
\(x \in Y_1\), we have \(x^\psi = x^\overline{\psi} m_x\) for some \(m_x \in M\), where by \(x^\overline{\psi}\) we mean the
word in \(Y_1\) corresponding to the action of \(\overline{\psi}\) on \(G/M\). The proposed map \(\psi\) is
then an automorphism of \(G\) if and only if the images of the generators satisfy
the relators of \(G\). These relators reduce to a system of linear equations for the
unknown elements \(m_x \in M\) over \(K = GF(p)\). In fact the matrix for this system
of equations is just the matrix \(E\) that we have already computed, as described in
subsection 5.1, and is the same for all \(\overline{\psi}\); it is only the vector on the right hand
side of the system of equations that has to be computed for each individual \(\overline{\psi}\).
If there is a solution, then we have found an inverse image \(\psi\), and if not, then
\(\overline{\psi} \notin \text{Im}(\rho)\).

Given that we have a procedure for deciding whether a specific \(\overline{\psi}\) lies in \(\text{Im}(\rho)\),
we now have to find the full subgroup \(\text{Im}(\rho)\) of \(A_M\). Since \(A_M\) may be large, we
certainly do not want to have to test every element individually. However, this is
not usually necessary. We can represent \(A_M\) as a permutation group, and carry
out a standard backtrack search for the required subgroup. Since the theory of
such searches has been described in detail elsewhere, we shall not repeat it here,
and refer the reader to Chapter 11 of Butler [1991]. Since \(\rho\) clearly maps \(BI\)
tonto \(I_M\), we can initialize the sought subgroup to \(I_M\), and then the set \(O_A\) of
automorphisms of \(G\) that we find whose images generate \(\text{Im}(\rho)\) modulo \(I_M\) will
generate \(A\) modulo its subgroup \(BI\).

Of course, we still have to find a suitable permutation representation of \(A_M\).
Currently we use one of two possibilities. If \(A_M\) is not too large, then we use
the regular representation of degree \(|A_M|\). If \(G/M\) is much smaller than \(A_M\),
however, then we can use the action of \(A_M\) on the non-identity elements of
\(G/M\), which has degree \(|M| - 1\). We would have problems if both \(A_M\) and \(G/M\)
were large (bigger than about 50,000, for example), but fortunately this does
not seem to occur in examples within the current range of the algorithm.

If we are testing groups \(H\) and \(G\) for isomorphism, then we already have an
isomorphism \(\xi_M : H/M^H \to G/M\) and, as explained in subsection 2.2, for each
candidate \(\overline{\psi} \in \text{Aut}(G/M)\) that does not lift to \(\psi \in \text{Aut}(G)\), we need to test
whether \(\xi_M \overline{\psi}\) lifts to an isomorphism \(\xi : H \to G\). If we complete the search
without finding such a lifting, then \(H\) and \(G\) are not isomorphic. If we do find a
lifting, then we still carry on and complete the computation of \(O_A\) for \(G\), because
we shall need this for the next factor in the characteristic series of \(G\). Of course,
if we are dealing with the final factor in the lifting process, then we could, if we
wanted to, abort the search as soon as we find a single lifting.

The lifting test for \(\xi_M \overline{\psi}\) is similar to that for \(\overline{\psi}\). First we check whether \(\xi_M \overline{\psi}\) in-
duces a module automorphism from the \(H/M^H\)-module \(M^H\) to the \(G/M\)-module
M. If not, then $\xi_M \psi$ does not lift. If it does, and we are in the split case, then we can immediately write down a lifting $\xi$ that induces $\xi_M$ on the given complement of $M^H$ in $H/M^H$. In the non-split case, for all module automorphisms of $M^H$ we have to test whether a certain system of linear equations, the matrix of which is the $E^H$ calculated in subsection 5.1, has a solution.

5.4. Finding a Presentation of $A$

We have now described how to find a set of generators $Y_\gamma \cup O_C \cup O_B \cup O_A$ of $A$, where $Y_\gamma$ generates $I$. These generating automorphisms are all defined by their action on the generating set $Y = Y_1 \cup Y_2$ of $G$. The methods described in the preceding three subsections enable us to calculate the order $|A|$ of $A$, but we also need to have a description of $A$ as a group. In particular this will be required if we need to form a permutation representation of $A$ in order to lift to the next layer. If we are testing two groups $H$ and $G$ for isomorphism, then we only need to carry out these calculations for $G$.

The most natural approach seems to be to define $A$ by finding defining relations on the generating set, such that the resulting presentation exhibits the series

$$1 \leq I \leq CI \leq BI \leq A.$$ 

This enables the regular or other suitable permutation representations of $A$ to be found easily (although, when $G$ is small, the action on the non-identity elements of $G$ can be found directly from the definitions of the generating automorphisms).

We get a set of defining relators of $I$ on the generators $Y_\gamma$, by simply taking the images of the defining relators of $G$, together with the image under $\gamma$ of the subgroup $Z$, which we found in subsection 5.1. We already know the actions of $O_C, O_B$ and $O_A$ on the generators $Y_\gamma$ of $I$, since this is how these automorphisms are defined.

We can find presentations of the three sections $CI/I \cong C/C \cap I, BI/CI \cong B/(B \cap CI)$ and $A/BI$ of $A$ as follows. The group $C$ is elementary abelian, and we have stored generators of $I_C$ of $C \cap I$ so this is unproblematic. The group $B/C$ has a representation as a matrix group, and we discussed the calculation of presentations of $B/C$ and of $B/(B \cap CI)$ in subsection 5.2. Finally, $A/BI$ is either the whole of $A_M/I_M$, in which case we have a presentation from the previous layer calculations, or we have a description of it as a subgroup of a permutation representation of $A_M/I_M$. We can find a presentation of a permutation group by using standard methods (see, for example, sections 5 and 7 of Neubüser [1982]), which may involve forming the regular representation for small groups, or using a strong generating set for larger groups. In the latter case, we may need to adjoin additional (redundant) generators to $O_A$ corresponding to strong generators of $A/BI$.

Starting from these three presentations, in order to find the relations and relators in a presentation of $A_M$ that exhibits the above series, we need to be able to transform a relator $w$ of $A/BI$, say, into a relation $w = v$ of $A$, where $v$
is a word in $Y^\gamma \cup O_C \cup O_B$ that has to be determined, and we need to be able to compute the conjugation action of generators in $O_A$ on those in $Y^\gamma \cup O_C \cup O_B$, etc. However, we can do all of these things provided that, for a given automorphism $\psi \in BI$, we know how to write $\psi$ as a word in $Y^\gamma \cup O_C \cup O_B$.

Our strategy here is to write $\psi = \psi_B \psi_C \psi_Y$, where $\psi_B, \psi_C$ and $\psi_Y$ are words in $O_B, O_C$ and $Y^\gamma$, respectively. Now, since $\psi \in BI$, the induced automorphism $\psi^\rho = \overline{\psi}$ of $G/M$ lies in $I_M$. The first step is find a word $w$ in $Y_1$ such that $w^\gamma$ induces $\overline{\psi}$. In other words we have to find a $w$ which induces a given conjugation action on $G/M$. We shall describe how to do this in subsection 5.5 below. We can then multiply $\psi$ (on the right) by $(w^\gamma)^{-1}$ to get $\psi \in B$.

At this stage, $\psi_B$ is determined by $\psi_C \in B/C$, so to find $\psi_B$, we merely have to write $\psi_C$ as a word in the matrix generators of $B/C$ (which are in one-one correspondence with the generators in $O_B$), and this can be done as a matrix group calculation; see the discussion in subsection 5.2 above. Having determined $\psi_B$, we can multiply $\psi$ on the left by $\psi_B^{-1}$ to get $\psi \in C$.

The group $C$ is an elementary abelian $p$-group, and $O_C$ generates $C$ modulo $C \cap I$. As described in subsection 5.1, finding a given automorphism in $C$ as a word in the generators of $C$ is elementary linear algebra. Now from subsection 5.1, our generating set of $C$ is equal to the union of $I_C$ and $O_C$ where, for each $\phi \in I_C$, we have stored a word $w \in Y$ with $w^\gamma = \phi$. Thus we can write $\psi$ as a product of words in $O_C$ and $Y^\gamma$, which completes the process.

### 5.5. Identifying a Given Inner Automorphism

The outstanding problem from subsection 5.4 was to find an element in $G/M$ (as a word in the generators of $G$ modulo $M$) whose conjugation action induces a given inner automorphism $\psi$ of $G/M$. We discussed this problem in subsection 2.4, but we cannot apply the solution described there directly, because we do not have a permutation representation of $G/M$.

Recall that $M$ is one of the characteristic subgroups $N_k$ in the series,

$$1 = N_r < N_{r-1} < \ldots < N_1 = L \leq G,$$

and that our generating set of $G$ is a disjoint union of sets $X_i$ ($1 \leq i \leq r$) with $X_i \subseteq N_{i-1} \setminus N_i$.

If $G/L$ is non-trivial, then we first find a word $w$ in $X_1$ whose conjugation action induces the same inner automorphism of $G/L$ as $\psi$. Since we do have a permutation representation of $G/L$ available, we can apply the method of subsection 2.4 to find a suitable conjugating element, and then write it as a word in $X_1$ as described in the final paragraph of subsection 3.2.

We can now multiply $\psi$ by $(w^\gamma)^{-1}$ and assume that $\psi$ acts trivially on $G/L$. In fact, inductively, we can assume that we have found a word $w$ in $X_1 \cup X_2 \ldots \cup X_{k-1}$ such that $\psi$ and $w^{\gamma N_{k-1}}$ induce the same action on $G/N_{k-1}$ and so we can assume that $\psi$ induces the identity on $G/N_{k-1}$. The problem then is to find a word $w$ in $X_1 \cup X_2 \ldots \cup X_k$ such that $\psi$ and $w^{\gamma N_k}$ induce the same action on $G/N_k$. 

Let $Y_1 = X_1 \cup X_2 \ldots \cup X_{k-1}$ and $Y_2 = X_k$, and let $B_k$, $C_k$ and $I_k$ be the subgroups corresponding to $B$, $C$ and $I$ for the section $N_{k-1}/N_k$ of $G$. Then the induced action of $\psi$ on $G/N_k$, which we shall also denote by $\psi$, is an element of $B_k \cap I_k$. Recall from subsection 5.2 that we have stored generators of this group as matrices, and we can use this matrix representation to find a word $v$ in our generating set $I_{B_k}$ of $B_k \cap I_k$ modulo $C_k \cap I_k$. Furthermore, for each $\phi \in I_{B_k}$, we have stored a word $w'$ in $Y_1$ with $(w')^{7k} = \phi$, and so we can compose these words $w'$ to form a word $w$ in $Y_1$ such that $w^{7k}$ and $\psi$ have the same restrictions to $N_{k-1}/N_k$.

Now, by multiplying $\psi$ by $(w^{7k})^{-1}$, we may assume that $\psi \in C_k \cap I_k$. Again, as described in subsection 5.1, we can find a word $w \in Y_1 \cap Y_2$ with $w^{7k} = \psi$, which completes the process.

6. Implementation Issues and Performance

Our current implementation of the automorphism group algorithm is written in the Magma language. It takes a finite permutation group $G$ as input and returns a group $A$ of automorphisms of $G$. Such a group belongs to a special Magma type called GrpAuto. The specialist $p$-group and soluble group automorphism functions also return groups belonging to this type. The elements of $A$ are automorphisms of $G$ and so their action on elements of $G$ can be immediately calculated. Elements of $A$ can also be composed and inverted, and their orders can be calculated. The order of $A$ itself is also stored as an attribute of $A$. In our implementation, an initial subset of the generators of $A$ generates $\text{Inn}(A)$, but this property is not guaranteed for all groups in GrpAuto. However, there is a general function which uses the method described in subsection 2.4 to determine whether or not a given automorphism is inner, and to find a corresponding conjugating element in $G$ when it is. Our implementation also returns a finitely presented group $F$ isomorphic to $A$, together with the isomorphism $F \to A$; the generators of $F$ are in one-one correspondence with those of $A$.

To carry out structural computations on a group $A$ of type GrpAuto, it is necessary to find a representation of $A$ as a permutation group, or as a PC-group if $A$ happens to be soluble. A function is provided that attempts to find a faithful permutation representation of $A$ acting on a suitable union of conjugacy classes of $G$. This will be effective whenever $G$ is not too large. The presentation returned by our implementation could also be used to attempt to find a permutation representation using coset enumeration, or to find a PC-presentation using the soluble quotient algorithm if $A$ is soluble.

In Sims [1997], Sims considered the problem of computing the order of subgroups of $\text{Aut}(G)$ defined by generating automorphisms supplied by the user, and of membership testing in such subgroups, and he described a GAP implementation of a proposed solution to these problems. He was particularly interested in the case when $G$ is a large finite $p$-group (for example, quotients of the Burnside group $B(2,5)$), where the automorphisms groups are large. Sims’ method also
uses a series of characteristic subgroups of $G$, and the automorphism groups of the associated factor groups. Although we have not done so yet, it would be straightforward to incorporate this idea into the authors’ implementation.

There are two potential bottlenecks in the performance, but only one of these seems really serious at present. The problem of expressing an element in a permutation representation of a trivial-Fitting group as a reasonably short word in a generating set starts to be noticeable when the group has order about 10,000. This can be to a large extent overcome by the use of strong generating sets. The normalizer calculations involved in computing automorphism groups of trivial-Fitting groups could also conceivably be slow in larger examples.

The most serious bottleneck can occur when calculating the image of $\rho : A/B \rightarrow A_M$ as described in subsection 5.3, particularly in the non-split case. Although we can hope for some improvements in performance here due to more efficient implementations, there will always remain difficult examples in which a search has to be carried out through a large group Aut($G/M$) for a relatively small subgroup of lifting automorphisms. Since the most difficult examples seem to be $p$-groups or groups with large sections that are $p$-groups, one way forward here might be to try to pass parts of the computation to the very efficient implementations that exist of O’Brien’s $p$-group automorphism algorithm [O’Brien, 1995]. We should emphasize that O’Brien’s algorithm uses more specialized methods and reduces the fundamental lifting problem to a subspace stabilizer problem in a group module, and it is orders of magnitude faster than our general purpose methods. In fact, recent work on the subspace stabilizer problem [Schwingel, 2000] is likely to result in further dramatic improvements to these methods in the near future. This problem is also discussed in section 5 of [Eick et al., 2002].

Table 1 contains some sample performance figures on a Sun Ultra 5 Work-Station with 1 Gigabyte of RAM, running under Solaris. The columns headed $|A|$ and $|O|$ list the orders of the automorphism group and outer automorphism group of the group $G$, respectively. The layer sizes are from the top downwards. All times are in seconds.

In general, the notation for groups follows that in Conway et al. [1985]. Thus $11_1^{1+2}$ and $11_0^{1+2}$ denote the two types of extraspecial groups of order $11^3$, where the former is the one with exponent 11. The group $\text{SL}_2(13) \lor Q_8$ is a central product of the two factors. The group $T(m,n)$ is the $n$-th transitive group of degree $m$, and is returned by the $\text{Magma}$ function $\text{TransitiveGroup}(m,n)$.

The examples $D_4^2$ and $D_6^5$ are indicative of some types of quite small groups for which the performance is unsatisfactory; the program will not complete in any reasonable time on $D_6^5$. This is because relatively few elements in the large automorphism group $\text{GL}(6,2)$ of the quotient of order $2^6$ lift to automorphisms of the full group, but the program attempts to decide which ones lift by means of a brute force search through $\text{GL}(6,2)$. These particular examples should not really be quite so difficult, because the automorphism group simply acts by permuting
the direct factors, and we hope to be able to improve the implementation to deal with this in the near future.

The four different timings for the group $T(16, 1671)$ were put in to give an indication of the effect that the degree of the permutation representation of the same group has on the performance. It appears from this and many other examples that we tried that there is virtually no deterioration in performance up to degree about 500, but thereafter we notice an increase in time that is rather less than linear in the degree.

A full catalogue of all transitive permutation groups of degree up to 30 has been computed by Hulpke [1996], and these provide convenient test examples. In fact we successfully computed the automorphism groups of all transitive permutation groups of degree up to 22, except for a few 2-groups of degree 16. The time for nearly all of these examples was less than one second. A summary of the performance is given in Table 2, where ‘hard examples’ means those that took more than 10 seconds, and 2-groups are excluded in degree 16. The hard examples in degree 20 are mostly extensions of an elementary abelian group of order $5^4$ by a 2-group with a fairly large top layer, such as $2^5$, and the time is being taken in testing whether automorphisms of this top layer lift.

| $G$                  | $|G|$  | $|A|$  | $|O|$  | Deg. | TF-Gp. | Layer Sizes | Time |
|----------------------|-------|-------|-------|------|--------|-------------|------|
| $11^{+12}$           | 1331  | 1597200 | 15200 | 121  | 1      | 121 11      | 2.5  |
| $11^6$               | 1331  | 13310  | 110   | 121  | 1      | 121 11      | 6.9  |
| $D_8 \times D_8$     | 64    | 2048   | 128   | 8    | 1      | 16 4        | 9.4  |
| $2^4 \times S_3$     | 128   | 3317760 | 51840 | 128  | 1      | 64 2        | 5886 |
| $D_6$                | 1296  | 31104  | 24    | 12   | 1      | 16 8        | 2.3  |
| $D_6$                | 7776  | 933120 | 120   | 15   | 1      | 32 243      | 718  |
| $2^4 \times SL_2(5)$ | 1920  | 23040  | 24    | 40   | $A_5$  | 60 2 16     | 0.8  |
| $2^8 A_5$            | 15360 | 88473600 | 5760  | 32   | $A_5$  | 60 256      | 0.6  |
| $C_3 \wr S_5$        | 29160 | 38880  | 4     | 15   | $S_5$  | 120 81 3    | 0.3  |
| $C_2 \wr C_{12}$     | 49152 | 12582912 | 512   | 24   | 1      | 4 2 3 16 8 8 2 | 20  |
| $AGL_2(7)$           | 98784 | 98784  | 1     | 49   | $PGL_2(7)$ | 336 3 2 49 | 0.9  |
| $ASL_2(49)$          | $2^5 \cdot 3^2 \cdot 7^6$ | $2^1 \cdot 3^2 \cdot 5^2 \cdot 7^6$ | 96    | 2401 | $L_2(49)$ | 58800 2 2401 | 59   |
| $SL_2(13) \times Q_8$| 17472 | 209664 | 48    | 64   | $L_2(13)$ | 1092 4 2 2 | 1.7  |
| $SL_2(13) \wr Q_8$   | 8736  | 52416  | 12    | 224  | $L_2(13)$ | 1092 4 2 2 | 1.6  |
| $3L_3(4)$            | 60480 | 241920 | 12    | 63   | $L_3(4)$ | 20160 3 2  | 1.7  |
| $3A_8 \times SL_2(5)$| 129600 | 172800 | 8     | 42   | $A_8 \times L_2(5)$ | 2160 3 2 | 1.5  |
| $A_5^2$              | 21600 | 10368000 | 48    | 15   | $A_5^3$ | 216000      | 0.6  |
| $A_5^6$              | 60    | 720 1206 | 46080 | 30   | $A_5^6$ | 60   6     | 39   |
| $SL_2(5)^6$          | 120$^6$ | 720 120$^6$ | 46080 | 30   | $A_5^6$ | 60   6     | 222  |
| $T(16, 1671)$        | 6144  | 36864  | 12    | 16   | 1      | 2 3 256 2 2 | 29   |
| $T(16, 1671)$        | 6144  | 36864  | 12    | 512  | 1      | 2 3 256 2 2 | 26   |
| $T(16, 1671)$        | 6144  | 36864  | 12    | 2048 | 1      | 2 3 256 2 2 | 96   |
| $T(16, 1671)$        | 6144  | 36864  | 12    | 6144 | 1      | 2 3 256 2 2 | 230  |
| $T(16, 1767)$        | 12288 | 98304  | 16    | 16   | 1      | 3 64 8 4 2 | 17   |
| $T(16, 1840)$        | 40320 | 40320  | 1     | 16   | $A_7$  | 2520 16     | 0.7  |
| $T(20, 893)$         | 320000 | 3840000 | 12    | 20   | 1      | 32 8 2 625 | 3727 |

Table 1: Times for some automorphism group calculations
<table>
<thead>
<tr>
<th>Degree</th>
<th>No. of Groups</th>
<th>No. of Hard Examples</th>
<th>Median Time</th>
<th>Longest Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
<td>0</td>
<td>0.02</td>
<td>0.03</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>0</td>
<td>0.04</td>
<td>0.21</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>0</td>
<td>0.05</td>
<td>0.23</td>
</tr>
<tr>
<td>6</td>
<td>16</td>
<td>0</td>
<td>0.11</td>
<td>0.27</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>0</td>
<td>0.16</td>
<td>0.28</td>
</tr>
<tr>
<td>8</td>
<td>50</td>
<td>0</td>
<td>0.32</td>
<td>5.89</td>
</tr>
<tr>
<td>9</td>
<td>34</td>
<td>0</td>
<td>0.25</td>
<td>0.51</td>
</tr>
<tr>
<td>10</td>
<td>45</td>
<td>0</td>
<td>0.24</td>
<td>0.80</td>
</tr>
<tr>
<td>11</td>
<td>8</td>
<td>0</td>
<td>0.16</td>
<td>0.79</td>
</tr>
<tr>
<td>12</td>
<td>309</td>
<td>0</td>
<td>0.48</td>
<td>5.19</td>
</tr>
<tr>
<td>13</td>
<td>9</td>
<td>0</td>
<td>0.20</td>
<td>1.44</td>
</tr>
<tr>
<td>14</td>
<td>63</td>
<td>0</td>
<td>0.38</td>
<td>2.24</td>
</tr>
<tr>
<td>15</td>
<td>104</td>
<td>0</td>
<td>0.51</td>
<td>4.01</td>
</tr>
<tr>
<td>16</td>
<td>527</td>
<td>4</td>
<td>1.27</td>
<td>56.82</td>
</tr>
<tr>
<td>17</td>
<td>10</td>
<td>0</td>
<td>0.42</td>
<td>5.51</td>
</tr>
<tr>
<td>18</td>
<td>983</td>
<td>3</td>
<td>1.09</td>
<td>16.58</td>
</tr>
<tr>
<td>19</td>
<td>8</td>
<td>2</td>
<td>0.27</td>
<td>14.50</td>
</tr>
<tr>
<td>20</td>
<td>1117</td>
<td>43</td>
<td>1.48</td>
<td>3910.56</td>
</tr>
<tr>
<td>21</td>
<td>164</td>
<td>8</td>
<td>0.96</td>
<td>89.93</td>
</tr>
<tr>
<td>22</td>
<td>59</td>
<td>4</td>
<td>0.99</td>
<td>154.96</td>
</tr>
</tbody>
</table>

Table 2: Times for automorphism groups of low degree transitive groups

The group $2^{1+6}$ with the long runtime is an instance of a group of smallest order (we believe) for which the performance of the algorithm is unsatisfactory. In fact the special purpose method for computing automorphism groups of finite $p$-groups defined by power-commutator presentations described in O’Brien [1995] completes in less than 3 seconds on this example, which certainly illustrates the point that it is better to use a special purpose algorithm for $p$-groups.

On the other hand, the special purpose method for soluble groups described in Smith [1995] does not perform consistently faster on the soluble examples; indeed, sometimes it is significantly slower. Here are some comparative timings for the soluble group algorithm on some of the examples above.

<table>
<thead>
<tr>
<th>$G$</th>
<th>$11_2^{1+2}$</th>
<th>$11_4^{1+2}$</th>
<th>$D_8 \times D_8$</th>
<th>$D_8^3$</th>
<th>$C_2 \wr C_{12}$</th>
<th>$T(16, 1671)$</th>
<th>$T(16, 1767)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>0.01</td>
<td>16.9</td>
<td>2833</td>
<td>2819</td>
<td>0.4</td>
<td>5572</td>
<td>190</td>
</tr>
</tbody>
</table>

It should be said, however, that this comparison is not altogether appropriate, because the soluble group algorithm is applicable to many groups of very large order, to which we cannot apply our general purpose methods, because these groups do not have suitable low degree permutation representations.

We have not listed any times for the isomorphism testing program, although we carried out a large number of tests. Typically, we took a copy of a group, possibly changed the degree of the representation, conjugated it by a random permutation in the symmetric group, and then re-defined it with randomly chosen generators.
The time for computing the isomorphism between the original group and the newly constructed group was never more than twice that of the corresponding automorphism group computation, and was usually not much more than that time itself. The time for confirming non-isomorphism was nearly always very short, but again never more than twice that of the corresponding automorphism group computation.

References


