Computing canonical labellings of finite solvable groups

Santiago Barrera Acevedo, Heiko Dietrich, and Max Horn

Abstract. We define a canonical labelling function on the class of finite solvable groups so that two such groups G and $\tilde{G}$ are isomorphic if and only if $\mathrm{can}(G) = \mathrm{can}(\tilde{G})$. Specifically, $\mathrm{can}(G)$ is a group presentation that describes a group isomorphic to G, and our description explains how to construct an isomorphism $G \to \mathrm{can}(G)$. Our approach is motivated by O'Brien's (1994) canonical presentations for finite p-groups and utilises ideas from group cohomology first described by Robinson (1982) and automorphism group algorithms developed by Smith (1994), Holt (2001), and others. We also discuss a proof-of-concept implementation for the computer algebra system GAP and comment on the major bottlenecks and open research questions.

1. Introduction

A *canonical labelling* is a way of assigning a *label* to a mathematical object such that two objects receive the same label if and only if they are equivalent in some sense. For example, two complex square matrices are *similar* if and only if they have the same Jordan Normal Form; two finite dimensional semisimple Lie algebras over the complex numbers are *isomorphic* if and only if they have the same Dynkin diagram; two permutations are *conjugate* in a symmetric group if and only if they have the same cycle type; and two connected, orientable, and compact surfaces are *homeomorphic* if and only if they have the same genus.

Canonical labellings are important because they reduce the often difficult problem of checking equivalence to the simpler task of comparing labels for equality: Instead of asking whether two objects are equivalent, one computes canonical labels and asks whether these are equal. Canonical labellings of finite graphs are an established application that plays a crucial role in contemporary algorithms

La Trobe University, Department of Mathematical and Physical Sciences, Bundoora, VIC, Australia

Monash University, School of Mathematics, Clayton, VIC, Australia

RPTU University Kaiserslautern-Landau, Germany

E-mail addresses: S.BarreraAcevedo@latrobe.edu.au, heiko.dietrich@monash.edu, mhorn@rptu.de.

2020 *Mathematics Subject Classification.* 20F16; 20-08; 20-04.

Key words. canonical labelling; solvable groups; canonical presentations; group isomorphism.

The authors thank Bettina Eick, Derek Holt, and Eamonn A. O'Brien for comments. Dietrich was supported by a Universities Australia Grant in the Australia-Germany Joint Research Cooperation Scheme (*The SmallGroups Library: New Classifications via Symbolic Computation*). Dietrich would like to thank the Isaac Newton Institute for Mathematical Sciences, Cambridge, for support and hospitality during the programme *Latest trends in algebra and geometry*, where some of the work on this paper was undertaken; this work was supported by EPSRC grant EP/V521929/1.

for solving the graph isomorphism problem, see [**24**]. Building on these ideas, recent work of Jefferson et al. [**23**] describes a framework for deciding the more general problem of whether two elements lie in the same orbit under the action of a permutation group. For a more general discussion of canonisation algorithms for combinatorial objects, we refer to Schweitzer and Wiebking [**29**]. Applied to groups, Wiebking [**32**] later demonstrated a canonical labelling that can be used to decide permutational isomorphisms.

In this paper, we focus on canonical labellings for finite solvable groups and their practical applications in computational group theory, in particular, implementations for computer algebra systems. Since canonical labellings for group isomorphism would solve the group isomorphism problem, which is a famous and notoriously difficult problem (see [**12**] and the references therein), one should not expect such a labelling to work effectively for all finite groups. Focus is therefore on special classes of groups; we give some examples. For the class of finite cyclic groups, the order of a group is a suitable canonical label. For the class of finite abelian groups, the *abelian invariants* of a group can be used as a canonical label; see [**17**, Theorem 9.13]. For the class of finite groups of size at most 2000 (except 1024), an example that is particularly relevant in practice is the "IdSmallGroup" label provided by the SmallGroups Library, see Besche et al. [**3**] and the implementations in GAP [**16**] and Magma [**4**]: If G is a group of size n at most 2000 (not 1024), IdSmallGroup$(G) = [n, i]$ if and only if G is isomorphic to the i-th group in the SmallGroups Library list of groups of size n. This illustrates the earlier observation that canonical labels are useful for reducing a list $G_1, \ldots, G_m$ of groups up to isomorphism: rather than performing up to $m(m-1)/2$ *potentially expensive* isomorphism tests $G_i \cong G_j$, one computes the canonical labels $\ell_1, \ldots, \ell_m$ of the groups and performs up to $m(m-1)/2$ much cheaper equality checks $\ell_i = \ell_j$. This can be done even more efficiently if the labels can be sorted, for example, when they are integers.

The examples of group labellings mentioned so far have in common that the labelling is determined by exhibiting sufficiently many isomorphism invariants to distinguish the groups. In particular, this approach does not produce explicit isomorphisms, which are often useful in practice. One noteworthy exception, and the main motivation for the present work, is provided by O'Brien's [**26**] canonical presentations for finite p-groups resulting from the p-group generation algorithm; see also [**17**, Section 9.4]. O'Brien has described an algorithm that takes as input a group G of prime power order and constructs a canonical group presentation $\mathrm{can}(G)$ and an isomorphism $G \to \mathrm{can}(G)$ such that two groups G and $\tilde{G}$ of prime power order are isomorphic if and only if $\mathrm{can}(G) = \mathrm{can}(\tilde{G})$, that is, if and only if the two presentations are *equal* when considered as strings of symbols. For example, $\mathrm{can}(G) = \langle g_1, \ldots, g_r \mid g_1^p, \ldots, g_r^p,\ g_j^{g_i} = g_j\ (i < j)\rangle$ is one way to define a canonical presentation for an elementary abelian p-group G of rank r. Since every finite nilpotent group is the direct product of groups of prime power order, the canonical presentations constructed in [**26**] can easily be extended to canonical presentations for finite nilpotent groups.

1.1. Main results. The aim of this work is to describe and compute canonical labellings for finite solvable groups; denote this class of groups by $\mathcal{S}$. The usual way to describe a finite solvable group in computational group theory is via a consistent polycyclic group presentation (see Section 2.1), and in practice we often assume that every group in $\mathcal{S}$ is given via such a presentation. This is not a serious restriction because effective methods exist to compute such a presentation for a finite solvable group G, assuming one can work effectively with G to begin with.

Our main results are the following; let $G, \tilde{G} \in \mathcal{S}$ be finite solvable groups.

(A) We describe a canonical presentation $\mathrm{can}(G)$ and prove that $G \cong \tilde{G}$ if and only if $\mathrm{can}(G) = \mathrm{can}(\tilde{G})$; see Definition 3.12 and Theorem 3.13.

(B) We describe the construction of an isomorphism $G \to \mathrm{can}(G)$; see Section 4.

(C) We introduce our GAP package `CanonicalPcPres` [**1**]; see Section 6.

Our GAP package can be considered a proof-of-concept implementation of our algorithm, and while it works well for many examples, there are serious bottlenecks that are all related to difficult orbit calculations. We comment on these bottlenecks and opportunities for future research in Section 5.

We note that the class of finite solvable groups is an important class of groups. It contains *many* groups, for example, every group of odd order is solvable by the famous Odd-Order Theorem. It is also a class of groups that allows effective computations, see [**17**, Chapter 8] and Section 2.1. Lastly, similar to finite p-groups, every finite solvable group can be described by iterated group extensions by elementary abelian modules; this structure is the basis for our inductive approach, see Section 2.3. The definition of canonical labellings for non-solvable groups will require different methods, cf. Hulpke's [**20**] work on perfect groups.

In Remark 2.1 we explain that it is in theory easy to define *some* canonical presentation for a finite solvable group, but the naive definition given there is difficult to compute and not practical. Our work is concerned with a more practical approach to constructing a canonical presentation together with an isomorphism.

1.2. General approach and structure of the paper. The definition and construction of our canonical presentation is rather technical, which is why this section provides a top-level description of our approach; to keep this concise, we refer to Section 2 for all unexplained definitions.

Let G be a finite solvable group. We assume that we can compute with G; this holds, for example, if G is given by a permutation representation, polycyclic presentation, or reasonably small matrix representation. We first construct a certain characteristic subgroup series $G = G_1 > G_2 > \ldots > G_{c+1} = \{1\}$ such that each section G_i/G_{i+1} is elementary abelian. Our approach to construct $\mathrm{can}(G)$ is to iteratively construct epimorphisms $G \to \mathrm{can}(G/G_{i+1})$. The starting point for this iteration is the elementary abelian group G_1/G_2, for which a canonical presentation is defined in the introduction. Constructing G_1/G_2 and an epimorphism $G \to \mathrm{can}(G_1/G_2)$ can be achieved by the usual abelianisation algorithm via computing abelian invariants. For the next step, we describe $\mathrm{can}(G/G_{i+1})$ as

a *canonical* extension of $\mathrm{can}(G/G_i)$ by a module isomorphic to G_i/G_{i+1}. Similar ideas have been used by Smith [**31**], Holt [**18**], Cannon and Holt [**7**], and Howden [**19**] to describe algorithms for computing the automorphism group of a finite (solvable) group, see also [**17**, Section 8.9]. Some of these ideas date back to work of Robinson [**28**] on group cohomology.

We provide some more details on this iteration. Assume an epimorphism from G to $\mathrm{can}(G/G_i)$ has already been constructed. This epimorphism induces an isomorphism $\pi\colon G/G_i \to \mathrm{can}(G/G_i)$. Since G/G_{i+1} is an extension of G/G_i by G_i/G_{i+1}, this allows us to describe G/G_{i+1} as an extension of $\mathrm{can}(G/G_i)$ by G_i/G_{i+1}. Recall that $\mathrm{can}(G/G_i)$ is given by a canonical polycyclic presentation, and it turns out that the extension we seek can be described by a certain polycyclic presentation that *extends* the one of $\mathrm{can}(G/G_i)$; we recall these details in Section 2.2. To make this presentation canonical, we have to solve two problems: first, we need to choose a canonical action of $\mathrm{can}(G/G_i)$ on the module G_i/G_{i+1}; this is described in Section 3.1. Once the module structure is fixed, we need to choose a canonical 2-cocycle that describes the extension; this is discussed in Section 3.2. Throughout, "canonical" can be interpreted as "deterministic" such that the presentation $\mathrm{can}(G/G_i)$ depends only on the isomorphism type of G for each i. Determining the canonical module structure and the canonical 2-cocycle requires orbit calculations, which constitute the main practical bottleneck in our approach. We describe these details in the main text.

The structure of the paper is as follows. In Section 2 we discuss the required technical background, including details on finite polycyclic groups and their polycyclic extensions. In Sections 3 and 4, we describe $\mathrm{can}(G)$ and the construction of an isomorphism $G \to \mathrm{can}(G)$, respectively. Section 5 discusses the main bottlenecks of our implementation and outlines ideas for future work. We conclude in Section 6 with a brief discussion of our implementation.

2. Technical background

We recall the required information on finite solvable groups. Most of this material can be found in the Handbook of Computational Group Theory [**17**].

2.1. Finite solvable groups. A finite solvable group G is polycyclic, that is, it admits a subnormal series

$$G = G_1 \geqslant G_2 \geqslant \ldots \geqslant G_{n+1} = \{1\}$$

such that each section G_i/G_{i+1} is a finite cyclic group. A corresponding polycyclic generating set is a sequence $S = (g_1, \ldots, g_n)$ of elements $g_1, \ldots, g_n \in G$ such that each $G_i/G_{i+1} = \langle g_i G_{i+1} \rangle$; the relative orders are $(r_1, \ldots, r_n)$ where each $r_i = |G_i/G_{i+1}|$. Every $g \in G$ can be uniquely written in the form $g = g_1^{e_1} \ldots g_n^{e_n}$ where each $e_i \in \{0, \ldots, r_i - 1\}$; this is the normal form of g with respect to S. By construction, $g_k^{r_k} = w_k(g_{k+1}, \ldots, g_n)$ and $g_j^{g_i} = w_{i,j}(g_{i+1}, \ldots, g_n)$

for each i, j, k with $i < j$, where w_k and $w_{i,j}$ are normalised words in the given arguments, that is, $w_k = g_{k+1}^{e_{k+1}} \dots g_n^{e_n}$ is a normal form, and similarly for $w_{i,j}$.

It follows that G is isomorphic to the group given by the polycyclic presentation

$$(1)\ \{g_1, \dots, g_n \mid g_k^{r_k} = w_k\ (k = 1, \dots, n),\ g_j^{g_i} = w_{i,j}\ (1 \leqslant i < j \leqslant n)\}.$$

By slight abuse of notation, we identify a group presentation with the group that it defines. We refer to [**17**, Section 8] for more details on polycyclic groups.

Remark 2.1. Every finite solvable group G of order $p_1^{e_1} \dots p_n^{e_n}$ (prime-power factorisation) has a polycyclic series whose sections are all of prime order, hence it can be represented by a polycyclic presentation with $n = e_1 + \dots + e_n$ generators. There are finitely many such presentations on generators $g_1, \dots, g_n$, and one could define $\mathrm{can}(G)$ to be that presentation that is *minimal* among all such polycyclic presentations with respect to some ordering on the strings that represent these presentations. This approach obviously does not yield a practical algorithm.

2.2. Extensions of polycyclic groups. Let G be a finite solvable group defined by the polycyclic presentation (1). Let M be a G-module, given by the polycyclic presentation with polycyclic generators $(h_1, \dots, h_m)$ and relative orders $(s_1, \dots, s_m)$:

$$(2)\qquad \begin{aligned} M = \langle\, h_1, \dots, h_m \mid\ & h_k^{s_k} = u_k\ (k = 1, \dots, m), \\ & h_j^{h_i} = h_j\ (1 \leqslant i < j \leqslant m)\ \rangle, \end{aligned}$$

where each $u_k = u_k(h_{k+1}, \dots, h_m)$ is a normal form. The G-action on M is specified by normal forms $a_{i,j} = a_{i,j}(h_1, \dots, h_m)$ in M such that $h_j^{g_i} = a_{i,j}$ for all g_i and h_j.

Let E be an extension of G by M, that is, a group that has a normal subgroup isomorphic to M with quotient group G. We identify M with a normal subgroup of E and $G = E/M$ via an epimorphism $\pi\colon E \to G$ with kernel M. Consider a transversal $\tau\colon G \to E$, that is, a map such that $\pi(\tau(g)) = g$ for all $g \in G$; we assume that τ is normalised, that is, $\tau(1) = 1$. Every element in E can be written as $\tau(g)a$ for unique $g \in G$ and $a \in M$. The transversal τ yields a map

$$\gamma\colon G \times G \to M, \quad (g, h) \mapsto \tau(gh)^{-1}\tau(g)\tau(h).$$

It is well-known that γ is a 2-cocycle in $Z^2(G, M)$ and that E is isomorphic to the group $E(\gamma)$ defined as

$$E(\gamma) = \{(g, a) \mid g \in G, a \in M\}$$

with multiplication $(g, a)(h, b) = (gh, a^h b\gamma(g, h))$. Specifically, an isomorphism $E \to E(\gamma)$ can be defined by $\tau(g)a \mapsto (g, a)$; see for example [**8**, Proposition 5.8]. This isomorphism also shows that E and $E(\gamma)$ are in fact *equivalent extensions* of G by M; see [**8**, p. 77].

It is clear that $E(\gamma)$ is polycyclic with polycyclic generators

$$((g_1, 1), \dots, (g_n, 1), (1, h_1), \dots, (1, h_m))$$

and relative orders $(r_1, \dots, r_n, s_1, \dots, s_m)$. For every g_k there is a normal form $t_k = t_k(h_1, \dots, h_m)$ such that $(g_k, 1)^{r_k} = (w_k, t_k)$; similarly, for $i < j$ there exists a normal form $t_{i,j} = t_{i,j}(h_1, \dots, h_m)$ such that $(g_j, 1)^{(g_i,1)} = (w_{i,j}, t_{i,j})$. We collect all these words t_k and $t_{i,j}$ in a sorted list, the *tail vector* defined by γ:

$$t = \mathrm{tail}(\gamma) = (t_1, \dots, t_n, t_{1,2}, t_{1,3}, \dots, t_{n-1,n});$$

note that t also depends on the chosen module structure (implicitly via γ).

By slight abuse of notation, $E(\gamma)$ is isomorphic to the group defined by the polycyclic presentation with polycyclic generating set $(g_1, \dots, g_n, h_1, \dots, h_m)$, relative orders $(r_1, \dots, r_n, s_1, \dots, s_m)$, and relations

$$\begin{aligned}
g_k^{r_k} &= w_k t_k && (k = 1, \dots, n),\\
g_j^{g_i} &= w_{i,j} t_{i,j} && (1 \leqslant i < j \leqslant n),\\
h_k^{s_k} &= u_k && (k = 1, \dots, m),\\
h_j^{h_i} &= h_j && (1 \leqslant i < j \leqslant m),\\
h_j^{g_i} &= a_{i,j} && (i = 1, \dots, n; j = 1, \dots, m).
\end{aligned} \tag{3}$$

We denote this presentation by

$$E(t) = E(\mathrm{tail}(\gamma)). \tag{4}$$

There is a natural isomorphism $E(\gamma) \to E(\mathrm{tail}(\gamma))$ that maps the generators of $E(\gamma)$ to the generators of $E(\mathrm{tail}(\gamma))$; for details we refer to [**17**, Section 8.7.2].

What follows is [**17**, Lemma 8.47]: the map that takes a 2-cocycle $\gamma \in Z^2(G, M)$ to $\mathrm{tail}(\gamma) \in M^{(n^2+n)/2}$ is a group homomorphism; we denote its image by $\mathcal{Z}(G, M)$. If γ lies in the kernel of this homomorphism, then $E(\gamma) = G \ltimes M$ is a split extension, so $\gamma \in B^2(G, M)$ is a 2-coboundary. It follows that the 2-cohomology group $H^2(G, M) = Z^2(G, M)/B^2(G, M)$ is isomorphic to $\mathcal{Z}(G, M)/\mathcal{B}(G, M)$, where $\mathcal{B}(G, M)$ is the group generated by $\mathrm{tail}(\gamma)$ with $\gamma \in B^2(G, M)$.

It is important for our purposes that $\mathcal{Z}(G, M)$ and $\mathcal{B}(G, M)$ can be computed without knowledge of $Z^2(G, M)$ and $B^2(G, M)$, see [**17**, Section 8.7.2].

2.3. Characteristic series. Our iterative construction of $\mathrm{can}(G)$ requires us to use a suitable subgroup series of G. Such a series is usually constructed by an iterative refinement, for example, starting with the derived series of G; see Example 2.4. We denote the process of obtaining such a series a *canonical filtration*.

Definition 2.2. A *canonical filtration* is a process that takes as input a finite solvable group G and outputs a subnormal subgroup series

$$G = C_1(G) \geqslant C_2(G) \geqslant \ldots \geqslant C_{c+1}(G) = \{1\},$$

where $c = c(G)$ depends on G, with the following properties:

(a) the section $C_1(G)/C_2(G)$ is nilpotent, and each section $C_i(G)/C_{i+1}(G)$, for $i \geqslant 2$, is elementary abelian of prime-power order;

(b) if $\tilde{G}$ is isomorphic to G, then $c(\tilde{G}) = c(G)$, and every isomorphism $\varphi\colon G \to \tilde{G}$ induces isomorphisms $C_i(G) \to C_i(\tilde{G})$ for all i.

For a fixed filtration, the resulting subgroup series for G is called the *canonical characteristic subgroup series for* G.

It follows from property (b) that each $C_i(G)$ is characteristic in G.

Example 2.3. For a finite p-group G, the lower exponent-p series is a canonical characteristic subgroup series for G. It is defined by $C_1(G) = G$ and $C_{i+1}(G) = [G, C_i(G)]C_i(G)^{[p]}$ for $i \geqslant 1$, where $C_i(G)^{[p]}$ is the subgroup of $C_i(G)$ generated by all p-th powers and $[G, C_i(G)]$ is the subgroup generated by all commutators $[a, b] = a^{-1}b^{-1}ab$ with $a \in G$ and $b \in C_i(G)$; see [**17**, p. 355]. This series is the fastest descending series of G with elementary abelian sections; it satisfies $C_i(G/C_j(G)) = C_i(G)/C_j(G)$ for all $i \leqslant j$.

Example 2.4. For a finite solvable group G that is not a p-group, we construct a canonical filtration as follows. We start with the derived series of G. We leave the first quotient as it is; this is the abelianisation G/G', and the construction of the abelian invariants of G allows us to define $\mathrm{can}(G/G')$ and construct an epimorphism $G \to \mathrm{can}(G/G')$. We refine every other section S such that new subsequent sections are isomorphic to the Sylow subgroups of S in ascending order of the primes. This yields a characteristic series in which every section (except possibly the first) is a group of prime power order. Each of these (abelian) p-group sections is now refined via their lower exponent-p series, see Example 2.3. This construction yields a canonical filtration. Modifications exist, for example, we can refine G/G' further to arrange that every section in the series is elementary abelian. Alternatively, we can also impose that $C_2(G)$ is the nilpotent residual of G, that is, the unique smallest normal subgroup of G with nilpotent quotient $G/C_2(G)$; this is useful in practice, see Section 6. Another practical approach is to describe G by a *special polycyclic generating set* and use the associated subgroup series as described in [**6**].

Throughout, we assume we have fixed a canonical filtration, for example, the one defined in Example 2.4, and refer to the resulting subgroup series as *the* canonical characteristic subgroup series of G. To simplify the notation, we usually denote this series by $G = G_1 \geqslant G_2 \geqslant \ldots \geqslant G_{c+1} = \{1\}$.

3. Canonical presentations

The aim of this section is to define, for a finite solvable group G, a polycyclic presentation $\mathrm{can}(G)$ such that $G \cong \tilde{G}$ if and only if $\mathrm{can}(G) = \mathrm{can}(\tilde{G})$. Our construction also yields an explicit isomorphism $\varphi\colon G \to \mathrm{can}(G)$, which we will discuss in Section 4.

Let $G = G_1 \geqslant G_2 \geqslant \ldots \geqslant G_{c+1} = \{1\}$ be the canonical characteristic series of G, see Section 2.3. We construct $\mathrm{can}(G)$ as an iterated extension by working up this characteristic series. We assume that $H = \mathrm{can}(G/G_i)$ and an epimorphism $\pi\colon G \to H$ are constructed, and we then construct $\mathrm{can}(G/G_{i+1})$ as the *canonical* extension of G/G_i by G_i/G_{i+1} that is isomorphic to G/G_{i+1}. For this we first

discuss how to fix a *canonical* module structure; the second step is then to fix a *canonical* 2-cocycle. We deal with these tasks in the subsequent sections.

3.1. Module structure. Let G be a finite solvable group with canonical characteristic series $G = G_1 \geqslant \ldots \geqslant G_{c+1} = \{1\}$. Let $\alpha\colon G \to \tilde{G}$ be an isomorphism and fix $k \in \{2, \ldots, c\}$. To simplify the notation, the induced isomorphisms $G/G_k \to \tilde{G}/\tilde{G}_k$ and $G_k/G_{k+1} \to \tilde{G}_k/\tilde{G}_{k+1}$ are also denoted by α.

Note that G/G_{k+1} is an extension of G/G_k by G_k/G_{k+1}, and G/G_{k+1} acts on G_k/G_{k+1} by conjugation yielding a *module structure* $G/G_{k+1} \to \mathrm{Aut}(G_k/G_{k+1})$. Since G_k/G_{k+1} is abelian, this action induces a well-defined module structure $G/G_k \to \mathrm{Aut}(G_k/G_{k+1})$. The next lemma compares this module structure with the one associated with the extension $\tilde{G}/\tilde{G}_{k+1}$.

LEMMA 3.1. *Let $\alpha\colon G \to \tilde{G}$ be an isomorphism. If $\sigma\colon G/G_k \to \mathrm{Aut}(G_k/G_{k+1})$ and $\tilde{\sigma}\colon \tilde{G}/\tilde{G}_k \to \mathrm{Aut}(\tilde{G}_k/\tilde{G}_{k+1})$ are the module structures induced by conjugation, then for all $g \in G$*

$$\tilde{\sigma}(\alpha(g)\tilde{G}_k) = \alpha \circ \sigma(gG_k) \circ \alpha^{-1}.$$

PROOF. The automorphism $\alpha\circ\sigma(gG_k)\circ\alpha^{-1} \in \mathrm{Aut}(\tilde{G}_k/\tilde{G}_{k+1})$ maps $\alpha(h)\tilde{G}_{k+1}$ to $\alpha(h^g)\tilde{G}_{k+1}$, which is equal to $\alpha(h)^{\alpha(g)}\tilde{G}_{k+1}$; the claim follows. □

DEFINITION 3.2. Let A and B be groups and $(u, v) \in \mathrm{Aut}(A) \times \mathrm{Aut}(B)$. Let $\kappa_v\colon \mathrm{Aut}(B) \to \mathrm{Aut}(B)$ be conjugation by v^{-1}. For a group homomorphism $\varepsilon\colon A \to \mathrm{Aut}(B)$ define $(u, v) \cdot \varepsilon = \kappa_v \circ \varepsilon \circ u^{-1}$, that is,

$$(u, v) \cdot \varepsilon :\ A \to \mathrm{Aut}(B), \quad a \mapsto v \circ \varepsilon(u^{-1}(a)) \circ v^{-1}.$$

This defines a group action of $\mathrm{Aut}(A) \times \mathrm{Aut}(B)$ on the set of all A-module structures on B.

Now let $H = \mathrm{can}(G/G_k) = \mathrm{can}(\tilde{G}/\tilde{G}_k)$ and $M = C_p^d$ (the elementary abelian group of size p^d) such that $M \cong G_k/G_{k+1}$. Choose isomorphisms

$$\begin{aligned} &\varphi\colon G/G_k \to H, && \beta\colon G_k/G_{k+1} \to M, \\ &\tilde{\varphi}\colon \tilde{G}/\tilde{G}_k \to H, && \tilde{\beta}\colon \tilde{G}_k/\tilde{G}_{k+1} \to M. \end{aligned} \tag{5}$$

These isomorphisms allow us to translate the module structures σ and $\tilde{\sigma}$ described in Lemma 3.1 to H-module structures on M; we denote these by $\varepsilon\colon H \to \mathrm{Aut}(M)$ and $\tilde{\varepsilon}\colon H \to \mathrm{Aut}(M)$: If $h \in H$, then

$$\varepsilon(h) = \beta \circ \sigma(\varphi^{-1}(h)) \circ \beta^{-1} \quad \text{and} \quad \tilde{\varepsilon}(h) = \tilde{\beta} \circ \tilde{\sigma}(\tilde{\varphi}^{-1}(h)) \circ \tilde{\beta}^{-1}. \tag{6}$$

The next result shows that the H-module structures on M induced by isomorphic groups lie in the same orbit under the action of $\mathrm{Aut}(H) \times \mathrm{Aut}(M)$ defined in Definition 3.2.

PROPOSITION 3.3. *If $\tilde{\varepsilon}$ and ε are as in* (6)*, then with the notation of Definition 3.2, the following hold.*

a) *There is $(u, v) \in \mathrm{Aut}(H) \times \mathrm{Aut}(M)$ such that $\tilde{\varepsilon} = (u, v) \cdot \varepsilon$.*

b) *If $(u, v) \in \mathrm{Aut}(H) \times \mathrm{Aut}(M)$, then $\hat{\varepsilon} = (u, v) \cdot \varepsilon$ is the H-module structure on M defined by some group $\hat{G}$ isomorphic to G.*

PROOF. a) Let $h \in H$ and recall from (6) that $\sigma(\varphi^{-1}(h)) = \beta^{-1} \circ \varepsilon(h) \circ \beta$. Now Lemma 3.1 yields

$$\alpha \circ (\beta^{-1} \circ \varepsilon(h) \circ \beta) \circ \alpha^{-1} = \alpha \circ \sigma(\varphi^{-1}(h)) \circ \alpha^{-1} = \tilde{\sigma}(\alpha(\varphi^{-1}(h))),$$

and therefore

$$\begin{aligned}\tilde{\varepsilon}(h) &= \tilde{\beta} \circ \tilde{\sigma}(\tilde{\varphi}^{-1}(h)) \circ \tilde{\beta}^{-1} \\ &= \tilde{\beta} \circ \tilde{\sigma}(\alpha(\alpha^{-1}(\tilde{\varphi}^{-1}(h)))) \circ \tilde{\beta}^{-1} \\ &= \tilde{\beta} \circ (\alpha \circ \sigma(\alpha^{-1}(\tilde{\varphi}^{-1}(h))) \circ \alpha^{-1}) \circ \tilde{\beta}^{-1} \\ &= (\tilde{\beta} \circ \alpha \circ \beta^{-1}) \circ \varepsilon[\varphi(\alpha^{-1}(\tilde{\varphi}^{-1}(h)))] \circ (\beta \circ \alpha^{-1} \circ \tilde{\beta}^{-1}).\end{aligned}$$

Note that $v = \tilde{\beta} \circ \alpha \circ \beta^{-1} \in \mathrm{Aut}(M)$ and $u = \tilde{\varphi} \circ \alpha \circ \varphi^{-1} \in \mathrm{Aut}(H)$, and so the claim follows from

$$\tilde{\varepsilon} = \kappa_v \circ \varepsilon \circ u^{-1} = (u, v) \cdot \varepsilon.$$

b) As before, let $\varepsilon\colon H \to \mathrm{Aut}(M)$ be the module structure induced by G with respect to isomorphisms φ and β. Let $(u, v) \in \mathrm{Aut}(H) \times \mathrm{Aut}(M)$ and define $\hat{\varepsilon} = \kappa_v \circ \varepsilon \circ u^{-1}$. Let $\hat{G} = G$ and let $\hat{\alpha}\colon G \to \hat{G}$ be the identity. Now define $\hat{\varphi} = u \circ \varphi$ and $\hat{\beta} = v \circ \beta$. The proof of part a) shows that $\hat{\varepsilon}\colon H \to \mathrm{Aut}(M)$ is defined by the group $\hat{G}$ and isomorphisms $\hat{\alpha}$, $\hat{\varphi}$, and $\hat{\beta}$. □

The module structure $\varepsilon\colon H \to \mathrm{Aut}(M)$ is a group homomorphism defined on the canonical group $H = \langle x_1, \ldots, x_m \mid \mathcal{R} \rangle$, so determined by the matrix tuple $T = (\varepsilon(x_1), \ldots, \varepsilon(x_m))$. It is straightforward to define an ordering on the set of all module structures, for example, by representing such a matrix tuple as a single long vector and then using the lexicographical ordering. With respect to a fixed ordering, we define the following.

DEFINITION 3.4. The *canonical H-module structure on M* is the minimal element in the orbit of ε under the action of $\mathrm{Aut}(H) \times \mathrm{Aut}(M)$.

REMARK 3.5. The group $\mathrm{Aut}(M)$ acts by matrix conjugation on the matrix tuples describing our module structures. Determining a canonical element in that orbit is the *canonical forms for matrix tuples* problem, see [**33**] for recent progress on the theoretical complexity of this problem. Computing the orbit of a matrix tuple under $\mathrm{Aut}(H) \times \mathrm{Aut}(M)$ is one of the major bottlenecks of our practical implementation; we describe some attempts to improve practicality in Section 5.

REMARK 3.6. Let $H, M, \varphi, \beta, \varepsilon$ be as in (5) and (6). Let $(u, v) \in \mathrm{Aut}(H) \times \mathrm{Aut}(M)$ such that $\hat{\varepsilon} = (u, v) \cdot \varepsilon$ is the canonical module structure. The proof of Proposition 3.3 shows that if the isomorphisms φ and β are replaced by $\hat{\varphi} = u \circ \varphi$ and $\hat{\beta} = v \circ \beta$, respectively, then the module structure ε will be replaced by $\hat{\varepsilon}$.

Thus, going forward, we always assume that the isomorphisms φ and β are chosen such that the induced H-module structure on M is the canonical one, denoted $\varepsilon\colon H \to \mathrm{Aut}(M)$.

3.2. Canonical extensions. We now describe the construction of the canonical presentation. As before, let G be a finite solvable group with canonical characteristic subgroup series $G = G_1 \geqslant \ldots \geqslant G_{c+1} = 1$ as defined in Section 2.3. We assume we have computed the group $H = \mathrm{can}(G/G_k)$ with epimorphism $\pi\colon G \to H$, and we want to construct $\mathrm{can}(G/G_{k+1})$ as the canonical extension of H by $M \cong G_k/G_{k+1}$ isomorphic to G/G_{k+1}. For this we first consider G_k/G_{k+1} as an H-module and arrange that this module structure is the canonical one, see Definition 3.4. Having fixed this module structure, we then define $\mathrm{can}(G/G_{k+1})$ as the extension defined by a canonical tail vector in $\mathcal{Z}(H, M)$; cf. Section 2.2. We explain this in more detail.

The canonical presentation for the abelian group $G/G_2 = G/G'$ can be defined as follows.

Definition 3.7. Let $A \cong C_e$ be a cyclic group and factorise $e = p_1 \ldots p_\ell$ with primes $p_1 \geqslant p_2 \geqslant \ldots \geqslant p_\ell$. Then

$$\mathrm{can}(A) = \langle g_1, \ldots, g_\ell \mid g_i^{p_i} = g_{i+1}\ (i = 1, \ldots, \ell - 1),\ g_\ell^{p_\ell},\ g_j^{g_i} = g_j\ (i < j)\rangle.$$

Let A be an abelian group with abelian invariants $[e_1, \ldots, e_m]$. Then A is isomorphic to $C_{e_1} \times \ldots \times C_{e_m}$, and $\mathrm{can}(A)$ is defined as the polycyclic presentation on generators $g_1, g_2, \ldots$ describing the direct product of $\mathrm{can}(C_{e_1}), \ldots, \mathrm{can}(C_{e_m})$.

Note that the abelian invariants of G can be computed by applying the well-known abelian quotient algorithm, see [**17**, Section 9.2]; this also produces an epimorphism $G \to \mathrm{can}(G/G') = \mathrm{can}(G/G_2)$. Our approach is to lift this iteratively to canonical presentations for the quotients G/G_k for $k = 3, 4, \ldots$, until we have determined $\mathrm{can}(G/G_{c+1}) = \mathrm{can}(G)$.

Notation. *In the following, we assume that $k \in \{3, \ldots, c\}$ and we have an epimorphism $\pi\colon G \to H$ with induced isomorphism $\varphi\colon G/G_k \to H$, defined by $gG_k \mapsto \pi(g)$. By construction, G_k/G_{k+1} is an elementary abelian G-module, say of size p^m. We choose a basis of this module and define an isomorphism*

$$\beta\colon G_k/G_{k+1} \to M \quad \textit{where} \quad M = \langle a_1, \ldots, a_m \mid a_1^p, \ldots, a_m^p, a_j^{a_i} = a_j\ (i < j)\rangle.$$

Following Remark 3.6, we assume that φ (hence π) and β are chosen such that the induced module structure $\varepsilon\colon H \to \mathrm{Aut}(M)$ is the canonical one. We fix this module structure in the following and for $b \in M$ and $h \in H$ abbreviate $b^h = b^{\varepsilon(h)}$.

Fix a normalised transversal $\tau\colon G/G_k \to G/G_{k+1}$ and define

$$\gamma'\colon G/G_k \times G/G_k \to G_k/G_{k+1}, \quad (a, b) \mapsto \tau(ab)^{-1}\tau(a)\tau(b).$$

Then G/G_{k+1} is isomorphic to the group

$$E(\gamma') = \{(a, b) \mid a \in G/G_k, b \in G_k/G_{k+1}\}$$

with $(a,b)(c,d) = (ac, b^c d\gamma'(a,c))$ via $G/G_{k+1} \to E(\gamma')$, $\tau(a)n \mapsto (a,n)$.

We now translate these constructions to H and M, and define

$$\gamma = \beta \circ \gamma' \circ (\varphi^{-1}, \varphi^{-1}) \colon H \times H \to M. \tag{7}$$

By construction, the following holds.

LEMMA 3.8. *The map γ is a 2-cocycle in $Z^2(H,M)$ that defines a group*

$$E(\gamma) = \{(g,a) : g \in H, a \in M\}$$

with multiplication $(g,a)(h,b) = (gh, a^h b\gamma(g,h))$. This group is isomorphic to G/G_{k+1} via $(g,a) \mapsto \tau(\varphi^{-1}(g))\beta^{-1}(a)$.

We identify M with $1 \times M \leqslant E(\gamma)$. Lemma 3.8 shows that $E(\gamma) \cong G/G_{k+1}$ and there is an isomorphism that maps $M \leqslant E(\gamma)$ to $G_k/G_{k+1} \leqslant G/G_{k+1}$. If we repeat the same construction for a group $\tilde{G}$ that is isomorphic to G, then we obtain a 2-cocycle $\delta \in Z^2(H,M)$ with

$$E(\delta) \cong \tilde{G}/\tilde{G}_{k+1} \cong G/G_{k+1} \cong E(\gamma),$$

and there is an isomorphism $E(\delta) \to E(\gamma)$ that maps $M \leqslant E(\delta)$ to $M \leqslant E(\gamma)$. The isomorphism problem of such extensions can be solved via the group of compatible pairs, which is defined as

$$\mathrm{CP}(H,M) = \{(\mu,\nu) \in \mathrm{Aut}(H) \times \mathrm{Aut}(M) \mid \forall h \in H, b \in M : \nu(b^h) = \nu(b)^{\mu(h)}\},$$

and its action on $Z^2(H,M)$ where $(\mu,\nu) \in \mathrm{CP}(H,M)$ acts on $\psi \in Z^2(H,M)$ via

$$(\mu,\nu) \cdot \psi = \nu \circ \psi \circ (\mu^{-1}, \mu^{-1}).$$

We refer to [**8**, Section 5.1.3] for more details on compatible pairs; see also [**17**, **28**, **31**]. In particular, it is known that there is an induced action on $H^2(H,M)$. For the next result, see [**8**, Proposition 5.13].

PROPOSITION 3.9. *Let $\gamma, \tilde{\gamma} \in Z^2(H,M)$. There is an isomorphism $E(\tilde{\gamma}) \to E(\gamma)$ that maps $M \leqslant E(\tilde{\gamma})$ to $M \leqslant E(\gamma)$ if and only if the classes $\tilde{\gamma} + B^2(H,M)$ and $\gamma + B^2(H,M)$ lie in the same orbit under the action of* $\mathrm{CP}(H,M)$.

REMARK 3.10. Note that $(\mu,\nu) \in \mathrm{CP}(H,M)$ if and only if $\varepsilon(h) = \nu^{-1} \circ \varepsilon(\mu(h)) \circ \nu$ for every $h \in H$, which is equivalent to $(\mu,\nu) \in \mathrm{Aut}(H) \times \mathrm{Aut}(M)$ stabilising the module structure $\varepsilon \colon H \to \mathrm{Aut}(M)$; thus, $\mathrm{CP}(H,M) = \mathrm{Stab}_{\mathrm{Aut}(H)\times\mathrm{Aut}(M)}(\varepsilon)$.

The next result shows that the cohomology class of γ depends only on the isomorphism type of G; this is an important property for our construction of $\mathrm{can}(G)$.

PROPOSITION 3.11. *With the previous notation, the construction of the 2-cocycle γ depends on the chosen transversal τ and the choice of isomorphisms φ and β. The following hold for the cohomology class $\gamma + B^2(H,M)$.*

a) *The $\mathrm{CP}(H,M)$-orbit of $\gamma + B^2(H,M)$ is independent of the choices of φ, β, τ.*
b) *The $\mathrm{CP}(H,M)$-orbit of $\gamma + B^2(H,M)$ depends only on the isomorphism type of G.*

PROOF. a) Choosing a different transversal for τ changes γ' (and so γ) by a 2-coboundary, see [**8**, Lemma 5.6], which does not affect the cohomology class $\gamma+B^2(H,M)$. If φ' and β' are different choices for φ and β (that do not change the H-module structure ε), then $\varphi' = \mu\circ\varphi$ and $\beta' = \nu\circ\beta$ for some $(\mu,\nu) \in \mathrm{Aut}(H)\times \mathrm{Aut}(M)$, and, in fact, $(\mu,\nu) \in \mathrm{CP}(H,M)$ by Remark 3.6. The construction in (7) with φ' and β' now yields a 2-cocycle γ' that satisfies $\gamma' = \nu\circ\gamma\circ(\mu^{-1},\mu^{-1})$, hence γ' and γ lie in the same $\mathrm{CP}(H,M)$-orbit, as claimed.

b) Let $\psi\colon G\to\tilde{G}$ be an isomorphism, with induced isomorphisms $\psi_i\colon G/G_i \to \tilde{G}/\tilde{G}_i$ and $\psi_i'\colon G_i/G_{i+1}\to \tilde{G}_i/\tilde{G}_{i+1}$. Let $\tilde{\varphi}\colon \tilde{G}/\tilde{G}_k \to H$ and $\tilde{\beta}\colon \tilde{G}_k/\tilde{G}_{k+1}\to M$ be isomorphisms such that the induced H-module structure on M is the canonical one. Let $\tilde{\tau}\colon \tilde{G}/\tilde{G}_k\to\tilde{G}/\tilde{G}_{k+1}$ be a transversal. These maps allow us to define a 2-cocycle $\tilde{\gamma}\in Z^2(H,M)$ as in (7). We now show that for G we can choose a specific transversal τ and isomorphisms φ and β such that the corresponding 2-cocycle γ equals $\tilde{\gamma}$. Recall from a) that we can change τ, φ, and β, without changing the $\mathrm{CP}(H,M)$-orbit. Specifically, for G we now consider the isomorphisms $\varphi = \tilde{\varphi}\circ\psi_k$ and $\beta = \tilde{\beta}\circ\psi_k'$, and transversal $\tau = \psi_{k+1}^{-1}\circ\tilde{\tau}\circ\psi_k$. We show with a direct calculation that these maps define the 2-cocycle $\gamma=\tilde{\gamma}$; let $a,b\in H$ and recall the definition of γ:

$$\begin{aligned}\gamma(a,b) &= \beta\big[\gamma'(\varphi^{-1}(a),\varphi^{-1}(b))\big]\\ &= \tilde{\beta}(\psi_k'\big[\gamma'(\,\psi_k^{-1}(\tilde{\varphi}^{-1}(a)),\ \psi_k^{-1}(\tilde{\varphi}^{-1}(b))\,)\big]).\end{aligned}$$

By definition, $\gamma'(u,v) = \tau(uv)^{-1}\tau(u)\tau(v)$ and $\tau(x) = \psi_{k+1}^{-1}(\tilde{\tau}(\psi_k(x)))$; recall also that $\psi_k'\circ\psi_{k+1}^{-1}(u) = u$ for all $u\in\tilde{G}/\tilde{G}_{k+1}$, and so

$$\begin{aligned}\gamma(a,b) &= \tilde{\beta}(\psi_k'\big[\psi_{k+1}^{-1}(\tilde{\tau}(\psi_k(\psi_k^{-1}(\tilde{\varphi}^{-1}(ab)))))^{-1}\\ &\qquad \psi_{k+1}^{-1}(\tilde{\tau}(\psi_k(\psi_k^{-1}(\tilde{\varphi}^{-1}(a)))))\\ &\qquad \psi_{k+1}^{-1}(\tilde{\tau}(\psi_k(\psi_k^{-1}(\tilde{\varphi}^{-1}(b)))))\big])\\ &= \tilde{\beta}(\,\tilde{\tau}(\tilde{\varphi}^{-1}(ab))^{-1}\tilde{\tau}(\tilde{\varphi}^{-1}(a))\tilde{\tau}(\tilde{\varphi}^{-1}(b))\,)\\ &= \tilde{\beta}(\,\tilde{\gamma}'(\tilde{\varphi}^{-1}(a),\tilde{\varphi}^{-1}(b)))\\ &= \tilde{\gamma}(a,b).\end{aligned}$$

It remains to show that φ and β define the same (canonical) H-module structure on M as $\tilde{\varphi}$ and $\tilde{\beta}$. If σ denotes the conjugation action of G/G_k on G_k/G_{k+1}, and $\tilde{\sigma}$ denotes the conjugation action of $\tilde{G}/\tilde{G}_k$ on $\tilde{G}_k/\tilde{G}_{k+1}$, then Lemma 3.1 shows that $\tilde{\sigma}(\psi_k(g)) = \psi_k'\circ\sigma(g)\circ\psi_k'^{-1}$ for all $g\in G/G_k$. Now (6) shows that the module action defined by $\tilde{\varphi}$ and $\tilde{\beta}$ maps $h\in H$ to

$$\tilde{\beta}\circ\tilde{\sigma}(\tilde{\varphi}^{-1}(h))\circ\tilde{\beta}^{-1} = \tilde{\beta}\circ\psi_k'\circ\sigma(\psi_k^{-1}(\tilde{\varphi}^{-1}(h)))\circ\psi_k'^{-1}\circ\tilde{\beta}^{-1}$$

which is the same as the module structure defined by $\varphi=\tilde{\varphi}\circ\psi_k$ and $\beta=\tilde{\beta}\circ\psi_k'$, which maps $h\in H$ to

$$\beta\circ\sigma(\varphi^{-1}(h))\circ\beta^{-1} = \tilde{\beta}\circ\psi_k'\circ\sigma(\psi_k^{-1}(\tilde{\varphi}^{-1}(h)))\circ\psi_k'^{-1}\circ\tilde{\beta}^{-1}.$$

This completes the proof. □

It follows from the previous discussion that the 2-cocycles in $Z^2(H, M)$ that define extensions isomorphic to G/G_{k+1} with M corresponding to G_k/G_{k+1} are exactly the 2-cocycles in the union of all cosets $\nu \circ \gamma \circ (\mu^{-1}, \mu^{-1}) + B^2(H, M)$ where (μ, ν) runs over the elements in $\mathrm{CP}(H, M)$. Moreover, this union only depends on the isomorphism type of G. It is a finite set of maps on finite sets, and therefore can be furnished with an ordering that allows us to pick a "minimal element" λ in a canonical way, see also Section 4. As canonical presentation for G/G_{k+1} we now set

$$\mathrm{can}(G/G_{k+1}) = E(\mathrm{tail}(\lambda)),$$

where $E(\mathrm{tail}(\lambda))$ is as defined in (4). We use this to define $\mathrm{can}(G)$ as follows.

Definition 3.12. Let G be a finite solvable group with canonical characteristic subgroup series $G = G_1 \geqslant \ldots \geqslant G_{c+1} = \{1\}$. Starting with $\mathrm{can}(G/G_2)$ as in Definition 3.7, we iterate the previous constructions of $\mathrm{can}(G/G_k)$ for $k = 2, 3, \ldots, c+1$, and set $\mathrm{can}(G) = \mathrm{can}(G/G_{c+1})$.

In each iteration, we first determine the canonical module structure and then the canonical 2-cocycle. The results discussed in this section imply the following.

Theorem 3.13. *If G and $\tilde{G}$ are finite solvable groups, then $G \cong \tilde{G}$ if and only if* $\mathrm{can}(G) = \mathrm{can}(\tilde{G})$.

Remark 3.14. We compare our approach with the computation of a canonical presentation for finite p-groups described in [**26**]. The approach in [**26**] also uses an iteration, namely, along the lower exponent-p central series. Assuming a canonical presentation for $H = G/P_k(G)$, it computes a canonical presentation of $G/P_{k+1}(G)$ by considering the p-cover H^* of H and constructing a canonical quotient of H^* that is isomorphic to $G/P_{k+1}(G)$. This canonical quotient depends on the choice of a certain canonical *allowable* subgroup of the p-multiplier of H, which also requires an orbit calculation. This algorithm is part of the ANUPQ algorithms (see [**15**] for the GAP implementations) that provide highly efficient implementations for p-group generation, p-quotient computation, and computations of canonical presentations. Our approach does not involve a covering group, but it computes a canonical presentation by considering canonical tail vectors describing the given group as suitable extensions. The *cover* constructions in [**10**] or [**13**] could potentially also be used to define a canonical presentation for finite solvable groups; however, these methods pose some additional technical challenges, which is why we have opted for the approach described in this work.

4. Constructing isomorphisms

We comment on a few of the crucial steps of the algorithm for computing $\mathrm{can}(G)$ and explain how to construct the isomorphism $G \to \mathrm{can}(G)$ along the way. We continue with the notation of the previous section, that is, we consider $G/G_{k+1} \cong E(\gamma)$ as an extension of the canonical presentation $H = \mathrm{can}(G/G_k)$ by the module $M \cong G_k/G_{k+1}$. The groups for the next iteration (which is necessary

if $G_{k+1} \neq \{1\}$) are $\tilde{H} = \mathrm{can}(G/G_{k+1})$ and $\tilde{M} \cong G_{k+1}/G_{k+2}$. We assume that isomorphisms are chosen such that the module structures are the canonical ones, see Definition 3.4 and Remark 3.6. Recall that $\tilde{H} = E(\mathrm{tail}(\lambda))$ for some $\lambda \in Z^2(H, M)$.

Tail vectors instead of cocycles. GAP provides functionality to compute the spaces $\mathcal{Z}(H, M)$ and $\mathcal{B}(H, M)$ of all tail vectors; see Section 2.2. Importantly, $H^2(H, M) \cong \mathcal{Z}(H, M)/\mathcal{B}(H, M)$, and there is a *deterministic* preimage function for the natural projection $\mathcal{Z}(H, M) \to \mathcal{Z}(H, M)/\mathcal{B}(H, M)$. After computing the tail vector $t \in \mathcal{Z}(H, M)$ corresponding to $\gamma \in Z^2(H, M)$, one can use GAP's functionality to enumerate the $\mathrm{CP}(H, M)$-orbit of the class $t + \mathcal{B}(H, M)$ in $\mathcal{Z}(H, M)/\mathcal{B}(H, M)$ and identify a canonical vector in that orbit (for example, the lexicographical least vector). We then take its (deterministic) preimage $\lambda \in \mathcal{Z}(H, M)$ to define $\mathrm{can}(G/G_{k+1}) = E(\mathrm{tail}(\lambda))$. In Section 5.2.1 we report on some ideas that aim to avoid the enumeration of the complete orbit, for example, in the case that M is a trivial H-module.

Iteration: the next epimorphism. For the next iteration, we need to construct an epimorphism $G \to \tilde{H}$. This can be done as follows. Continuing with the notation used above, Lemma 3.8 yields an isomorphism

$$G/G_{k+1} \to E(\gamma), \quad \tau(g)a \mapsto (\varphi(g), \beta(a)).$$

By construction, γ and λ define cohomology classes in the same $\mathrm{CP}(H, M)$-orbit, that is, there exists $(\mu, \nu) \in \mathrm{CP}(H, M)$ and $\delta_f \in B^2(H, M)$ such that

$$\nu \circ \gamma \circ (\mu^{-1}, \mu^{-1}) = \lambda + \delta_f,$$

where δ_f denotes the 2-coboundary defined by the map $f\colon H \to M$, that is, $\delta_f(g, h) = f(gh)^{-1} f(g)^h f(h)$ for $g, h \in H$. A short calculation confirms that this induces an isomorphism

$$E(\gamma) \to E(\lambda), \quad (g, a) \to (\mu(g), \nu(a) f(\mu(g))), \tag{8}$$

see also [**8**, Proposition 5.13]. Putting everything together, this yields an isomorphism

$$G/G_{k+1} \to E(\lambda), \quad \tau(g)a \mapsto (\mu(\varphi(g)), \nu(\beta(a)) f(\mu(\varphi(g)))),$$

and hence an induced epimorphism $G \to E(\lambda)$. Composing with the natural isomorphism from $E(\lambda)$ to $\tilde{H} = E(\mathrm{tail}(\lambda))$, see the discussion after (4), yields the required epimorphism $G \to \tilde{H}$. The construction of δ_f is explained in the next paragraph.

Iteration: the 2-coboundary. The orbit-stabiliser calculation of $\mathrm{CP}(H, M)$ on $\gamma + B^2(H, M)$ yields our chosen minimal class $\lambda + B^2(H, M)$, the stabiliser

$$S = \mathrm{Stab}_{\mathrm{CP}(H,M)}(\gamma + B^2(H, M)), \tag{9}$$

and an element $(\mu, \nu) \in \mathrm{CP}(H, M)$ such that $\delta_f = \nu \circ \gamma \circ (\mu^{-1}, \mu^{-1}) - \lambda$ is a 2-coboundary. In (8), we need the values $f(h_1), \ldots, f(h_n)$ to construct the

isomorphism $E(\gamma) \to E(\lambda)$. We do not know f, but we can compute values $\tilde{f}(h_1), \ldots, \tilde{f}(h_n)$ of some function $\tilde{f}\colon H \to M$ that satisfies $\delta_f = \delta_{\tilde{f}}$; this can be done by solving a linear equation system (cf. [**17**, p. 251]), and these values can be used to construct the required isomorphism.

Iteration: the next automorphism group. We also need to compute the automorphism group of $\tilde{H}$, and our algorithm provides most of the ingredients that are necessary for that; this follows the approach of [**18**, **31**]. For this we first recall how to construct generators for $\mathrm{Aut}(E(\lambda))$, and then use the isomorphism $E(\lambda) \to E(\mathrm{tail}(\lambda)) = \tilde{H}$ to translate these to generators for $\mathrm{Aut}(\tilde{H})$. Let S be as in (9). Generators for $\mathrm{Aut}(E(\lambda))$ come in two types. The first type can be constructed by taking automorphisms $(h, a) \to (\alpha(h), \beta(a)\iota(h))$ defined by compatible pairs $(\alpha, \beta) \in S$ stabilising $\lambda + B^2(H, M)$, where $\iota\colon H \to M$ is a map satisfying $\beta \circ \lambda \circ (\alpha^{-1}, \alpha^{-1}) - \lambda = \delta_\iota$. The second type can be constructed by taking automorphisms $(h, a) \to (h, a\kappa(h))$ for $\kappa \in Z^1(H, M)$, see [**8**, Proposition 5.14]. Note that (μ, ν) conjugates S to $\mathrm{Stab}_{\mathrm{CP}(H,M)}(\lambda + B^2(H, M))$ and that $Z^1(H, M)$ consists of all maps $\kappa\colon H \to M$ with $\kappa(1) = 1$ and $\kappa(ab) = \kappa(a)^b\kappa(b)$ for all $a, b \in H$, see [**8**, p. 74]; GAP provides tools to construct $Z^1(H, M)$.

5. Main bottlenecks and open research questions

The major bottlenecks of our implementation are the following tasks, which are all related to orbit-stabiliser calculations and often depend on the rank of the module M, the size of $\mathrm{Aut}(H)$, and the dimension of the space $H^2(H, M)$.

(T1) Computing the canonical module structure: this is an orbit calculation of $\mathrm{Aut}(H) \times \mathrm{Aut}(M)$ on matrix tuples, see Remark 3.5.

(T2) Computing $\mathrm{CP}(H, M)$: this is a stabiliser calculation which can be carried out along the orbit calculation in (T1), see Remark 3.10.

(T3) Computing the canonical 2-cocycle: this is an orbit calculation of the group $\mathrm{CP}(H, M)$ acting on $H^2(H, M)$.

As described in Section 4, in order to construct our isomorphism, in the orbit calculations during (T1) and (T3) we not only have to identify a minimal element in the orbit, but we also require a transversal element that maps the given element to the minimal one. Often we also seek a generating set for the stabiliser; this is relevant, for example, in (T2) or for the computation of the automorphism group.

Each of these tasks (T1), (T2), and (T3) is an interesting and challenging research problem in its own right; e.g., recall that (T1) encompasses the matrix tuple equivalence problem [**33**], see Remark 3.5. We comment on (T1) and (T2) in Section 5.1, and on (T3) in Section 5.2. We highlight these problems here to indicate scope for future research; any progress on these problems will be incorporated in future versions of our implementation [**1**]. Another opportunity for future improvements is to exploit the characteristic structure of groups; this is already used for

computing automorphism groups of finite p-groups, but not yet for isomorphism construction; see also the recent work [**5**].

We note that GAP already provides some advanced functionality for computing with large orbits, see for example the GAP package Orb [**25**]. Also relevant is an algorithm of Schwingel [**30**] that allows one to construct a canonical G-orbit representative for a p-subgroup $G \leqslant \mathrm{GL}_d(p)$ acting on the natural module, including stabiliser and transversal element; see [**14**, Section 5.2] for details. In the introduction we mentioned the work [**23**] that discusses canonical elements in orbits; it assumes permutation groups as acting groups, and therefore is not directly applicable to our situation.

5.1. Computing the canonical module structure and compatible pairs. The module structure $\varepsilon\colon H \to \mathrm{Aut}(M)$ is a homomorphism on the canonical group $H = \langle x_1, \ldots, x_m \mid \mathcal{R}\rangle$, so determined by the tuple $T = (\varepsilon(x_1), \ldots, \varepsilon(x_m))$. If all $\varepsilon(x_i)$ are scalar matrices, then we define $\mathrm{can}_M(T) = T$. Otherwise, let $j \geqslant 1$ be the first index such that $\varepsilon(x_j)$ is not scalar. We then define $\mathrm{can}_M(T)$ as the tuple $(M_1, \ldots, M_m)$, where $M_i = \varepsilon(x_i)$ for $i = 1, \ldots, j-1$, $M_j = B^{-1}\varepsilon(x_j)B$ is the Frobenius Normal Form of $\varepsilon(x_j)$, and $(M_{j+1}, \ldots, M_m)$ is the unique minimal element in the $C_{\mathrm{Aut}(M)}(M_j)$-orbit of $(B^{-1}\varepsilon(x_{j+1})B, \ldots, B^{-1}\varepsilon(x_m)B)$; the latter can also be constructed by an iteration over the entries of that tuple, but we skip these details here. This computation also allows us to construct generators of the stabiliser, which yields a generating set for the compatible pairs, see (T2).

5.2. Computing the canonical 2-cocycle. Here we consider the action of $\mathrm{CP}(H, M)$ on the cohomology group $H^2(H, M)$. GAP already provides functionality to compute this action via a matrix representation of $\mathrm{CP}(H, M)$, and this computation can be made more efficient by employing various improvements. For example, one can compute and exploit the structure of a composition series of the module, or of a composition series of the group (and apply the aforementioned algorithm of Schwingel to a normal p-subgroup); we refer to [**9**] for a further discussion of this problem. Here we discuss another simplification, namely, for the case that the H-module structure on M is trivial.

5.2.1. *Trivial module structure: a reduction using linear algebra.* Suppose M is a trivial H-module, so $\mathrm{CP}(H, M) = \mathrm{Aut}(H) \times \mathrm{Aut}(M)$. In (T3), instead of enumerating the union of the $\mathrm{CP}(H, M)$-orbit of $t + \mathcal{B}(H, M)$, we exploit that the action of $\mathrm{Aut}(M) = \mathrm{GL}_d(p)$ is well-understood: we now describe how to solve (T3) by acting with $\mathrm{Aut}(H)$ on canonical $\mathrm{Aut}(M)$-orbit representatives.

Let t be a tail vector with entries $t_1, \ldots, t_m \in \mathbb{Z}_p^d$. The action of $A \in \mathrm{Aut}(M)$ on t is by component-wise multiplication $t_i \mapsto t_i A$. Thus, if we consider t as a matrix with rows $t_1, \ldots, t_m$, the task is to define a canonical element in the set $t \cdot \mathrm{Aut}(M)$. For this we determine integers $i_1 < \ldots < i_r$, each as small as possible, such that the set $\{t_{i_1}, \ldots, t_{i_r}\}$ spans the row space of t, where r is the rank of t. We then extend this set deterministically to a basis of $\mathbb{Z}_p^d$ (for example, by adding standard basis vectors $e_{j_1}, \ldots, e_{j_{d-r}}$ with $j_1 < \cdots < j_{d-r}$, each as small as possible) and let A be the inverse of the matrix that has this basis as row

vectors. Now define $\mathrm{can}_M(t)$ as the tail vector represented by the matrix tA; we also call it the M-canonical tail vector of t. Note that tA has the property that it has the s-th standard vector e_s in row i_s for $s = 1, \ldots, r$. The next lemma shows that $\mathrm{can}_M(t)$ is indeed a canonical element in the set $t \cdot \mathrm{Aut}(M)$.

LEMMA 5.1. *Two tail vectors t and t' lie in the same $\mathrm{Aut}(M)$-orbit if and only if $\mathrm{can}_M(t) = \mathrm{can}_M(t')$.*

PROOF. By construction, if $\mathrm{can}_M(t) = \mathrm{can}_M(t')$, then $tA = t'A'$ for some $A, A' \in \mathrm{GL}_d(p)$, and therefore $t = t'A'A^{-1}$ and t' lie in the same $\mathrm{Aut}(M)$-orbit.

For the converse, suppose $t' = tA$ for some $A \in \mathrm{Aut}(M)$. Let $i_1 < \ldots < i_r$ be as in the construction of $\mathrm{can}_M(t)$. Since A is invertible, the vectors $t_{i_1}A, \ldots, t_{i_r}A$ span the row space of tA, hence of t'. By the same argument, if $i'_1 < \ldots < i'_r$ are as in the construction of $\mathrm{can}_M(t')$, then $t'_{i'_1}A^{-1}, \ldots, t'_{i'_r}A^{-1}$ forms a basis of the row space of t. By the minimality of the labels $i_1, \ldots, i_r$ and $i'_1, \ldots, i'_r$, we must have $i'_s = i_s$ for all $s = 1, \ldots, r$. Thus, $\mathrm{can}_M(t')$ and $\mathrm{can}_M(t)$ both have $e_1, \ldots, e_r$ in rows $i_1, \ldots, i_r$. Note that $\mathrm{can}_M(t)$ and $\mathrm{can}_M(t')$ lie in the $\mathrm{Aut}(M)$-orbit of t, so there is $B \in \mathrm{Aut}(M)$ with $\mathrm{can}_M(t') = \mathrm{can}_M(t)B$. Since $\mathrm{can}_M(t')$ and $\mathrm{can}_M(t)$ have the same rows $i_1, \ldots, i_r$, namely, the standard basis vectors $e_1, \ldots, e_r$, it follows that B fixes this basis of the row space of $\mathrm{can}_M(t)$ element-wise. This implies $\mathrm{can}_M(t') = \mathrm{can}_M(t)B = \mathrm{can}_M(t)$, as claimed. □

The $(\mathrm{Aut}(H) \times \mathrm{Aut}(M))$-action on tail vectors can now be reduced to an action of $\mathrm{Aut}(H)$ on the set of M-canonical tail vectors: given $t' = \mathrm{can}_M(t)$ and an automorphism $\alpha \in \mathrm{Aut}(H)$, we act with $(\alpha, 1) \in \mathrm{CP}(H, M)$ on t' as usual to obtain a tail vector t'', and then return $\mathrm{can}_M(t'')$ as the image of t' under α.

As mentioned in Section 4, in practice we act not on tail vectors but on a space that represents $H^2(H, M)$. We compute the M-canonical vector of $t \in H^2(H, M)$ as $\pi(\mathrm{can}_M(\pi^{-1}(t)))$, where $\pi\colon \mathcal{Z}(H, M) \to H^2(H, M)$ is the natural epimorphism. This is possible because in GAP the preimage function of π is deterministic.

6. Implementation and examples

We have implemented our algorithm for the computer algebra system GAP [**16**] in the package `CanonicalPcPres` [**1**]. In practice, for our canonical filtration we always impose that the first subgroup G_2 is the nilpotent residual of G, that is, the unique smallest normal subgroup of G with nilpotent quotient G/G_2. Since every finite nilpotent group is the direct product of its Sylow subgroups, this allows us to define and compute $\mathrm{can}(G/G_2)$ by using the canonical presentations for groups of prime power order, see [**26**]. Our current implementation therefore focuses on the general solvable case and provides the framework for constructing canonical extensions for not necessarily trivial module structure.

Before we present some examples, we need to set the expectations regarding runtimes. (All runtimes presented below have been obtained on a computer with

an Apple M4 CPU and 32GB RAM, using GAP 4.15.1.) Even the highly specialised ANUPQ algorithms with more than 22000 lines of C code [**27**, p. 3] currently require about 24 seconds to compute the standard presentation of the group $C_{11} \ltimes (C_{11} \ltimes (C_{11} \times C_{11^2}))$ with IdSmallGroup $[11^5, 86]$; this has been computed via the command `EpimorphismStandardPresentation` provided by [**15**]. (We note that more bespoke and efficient algorithms for dealing with groups of order dividing p^5 are currently developed in [**9**]; this is another reason why in our example computations below we focus on non-nilpotent groups.) The difficulty of computing canonical presentations is not so much determined by the size of the group, but heavily influenced by the largest module rank and, consequently, orbit sizes. Our algorithm faces the same challenges, and so there are already *small* orders where our algorithm requires a long time, but there are also *large* orders where we can compute the canonical presentation quickly; see Section 6.1 for explicit runtimes.

The purpose of our implementation is to provide the necessary framework for our algorithm; if more efficient solutions and implementations for the bottlenecks become available, then these will be incorporated and made public via the online repository [**1**] of our package. In this regard, we emphasise the following.

Remark 6.1. Our canonical presentation is *canonical* only in the sense that our algorithm produces a group presentation depending solely on the isomorphism type of the input group; this is achieved by selecting canonical orbit representatives with respect to some deterministic algorithm. If new orbit methods are implemented in future versions of our package [**1**], then this might change the choice of canonical representative, hence the canonical presentation. For this reason, when comparing (or referring to) canonical presentations, these should be computed using the same release version of [**1**].

We conclude with a brief remark on the natural question whether our algorithm makes any advances for the complexity of group isomorphism testing. The short answer is 'no'.

Remark 6.2. Investigating the complexity of group isomorphism testing is a famous problem in group theory and theoretical computer science. While in practice one is interested in algorithms that run in time polynomial in the *input size* (usually a generating set of size $O(\log n)$ for groups of size n), it is not even known whether the isomorphism problem can be solved in time polynomial in the *group size*; we refer to [**12**] for some recent results and references. The main bottleneck is expected to be isomorphism testing of finite p-groups; indeed, not even the case of p-groups of class 2 is known to be polynomial in the group size, see [**21**] and the references therein. The algorithm for canonical presentations of p-groups in [**26**] does not improve this complexity, so neither does our algorithm.

6.1. Examples. The command `IsomorphismCanonicalPcGroup` provided by our GAP package takes as input a finite solvable group G given by a polycyclic presentation (a GAP object of type `pc group`) and returns an isomorphism $G \to \mathrm{can}(G)$ whose image is the canonical presentation (`pc group`) of G; the function `CanonicalPcGroup` only returns $\mathrm{can}(G)$.

In Figure 1 we consider two 'small' groups; the first has IdSmallGroup $[1800, 430]$, and it takes a few milliseconds to compute the canonical presentation and isomorphism. To compare canonical presentations efficiently, we use the GAP function `CodePcGroup` to encode each presentation by an integer (*code*): the code together with the order of the group uniquely determines a polycyclic presentation of that group; we refer to [**2**, Section 3.3] or [**22**, Section 6.1] for more details. In Figure 1, we compute 1000 'random' copies of the group and illustrate that the canonical presentations are indeed all the same. Here and below we construct these random copies with the GAP command `RandomSpecialPcgsCoded`: it returns a group given by a *random* special polycyclic generating set [**6**], which provides particularly efficient group arithmetic.

The second group considered in Figure 1 has IdSmallGroup $[1296, 3583]$. It has a large-rank elementary abelian normal subgroup, and here it takes about 7 seconds to compute the canonical presentation and isomorphism. (While this is slower than hoped for, compare it with the performance of [**26**] for the group with IdSmallGroup $[11^5, 86]$; see above.) We present this example to illustrate that the bottlenecks are real. However, the examples below show that our implementation works very well in some other cases.

We used the SmallGroups Library to run over all solvable (but non-nilpotent) groups of size at most 2000 whose size is not divisible by any p^5, p a prime. There are 33450 such groups, and computing the canonical presentation and isomorphism takes on average less than 40ms. The most challenging size is $2^4.3^4$, see the group in Figure 1; on average each of the 3399 solvable but non-nilpotent groups of size $2^4.3^4$ can be done in 82ms. We also considered all solvable cubefree groups of size between 2001 and 50000; these groups are provided by the SmallGroups Library. There are 386623 such groups, and computing the canonical presentation and isomorphism takes on average less than 30ms. The longest runtime was 3 seconds for the group with IdSmallGroup $[36300, 489]$; however, on average, each of the 968 solvable groups of size 36300 is done in 70ms.

In Figure 2, we run over the 3087 solvable groups of cubefree size 44100. The cubefree groups of size at most 50000 are part of the SmallGroups Library. It took our code 219 seconds to compute all canonical presentations and isomorphisms (on average, 71ms per group). GAP's `IdSmallGroup` required 105 seconds to identify these 3087 groups, but our algorithm does more as it also provides explicit isomorphisms. GAP's `IsomorphismGroups` required close to 6 hours to perform these computations. The isomorphism algorithm for cubefree groups provided by the package Cubefree (see [**11**]) required 113 seconds to construct isomorphisms, but it does not provide a canonical labelling. In this example, our proof-of-concept implementation is already on par with `IdSmallGroup` and the bespoke command `IsomorphismCubefreeGroups`.

In Figure 3, we consider three larger groups G, K, and L, each defined by its code and size. The group G has cubefree size 2375789332049525700 and derived length 2; our implementation required less than one second to compute

the canonical presentation and isomorphism. The group K has size $2^7.3^5.5^4.7.11$ and derived length 4; our implementation took about 3 seconds to compute the canonical presentation and isomorphism. The group L has derived length 5 and size $2^{10}.3^7.5^2$; here our implementation took about 6 seconds.

```
gap> G := SmallGroup(1800,430);;  # 1800 = 2^3.3^2.5^2; 749 groups
gap> StructureDescription(G);
"C3 x (C3 : ((C5 x C5) : (C4 x C2)))"
gap> isoG := IsomorphismCanonicalPcGroup(G); time;
[ f1*f2,f3,f2*f4,f4,f5,f6^2*f7^3,f6^2*f7^2 ] -> [ f1,f2,f3,f4,f5,f6,f7 ]
56 # runtime in ms
gap> ### compute 1000 random copies and compare the canonical presentations
gap> code := CodePcGroup(CanonicalPcGroup(G)); # please note Remark 6.1
374453170123379004775891797023
gap> grps := List([1..1000],
                  x -> PcGroupCode(RandomSpecialPcgsCoded(G),Size(G)));;
gap> ForAll(grps, H -> code = CodePcGroup( CanonicalPcGroup(H) ) );
true
gap> ### a small order where the bottlenecks are visible
gap> K := SmallGroup(1296,3583);; # 1296 = 2^4.3^4; 3609 groups
gap> StructureDescription(K);
"C2 x (((C3 x C3) : C2) x S3 x S3)"
gap> CanonicalPcGroup(K);; time;
6661 # runtime in ms
```

FIGURE 1. Example computation for two small groups

```
gap> all  := AllSmallGroups(44100,IsSolvableGroup);; Size(all);
3087 # there are 3087 solvable groups of size 44100 = 2^2.3^2.5^2.7^2
gap> ### create two identical lists of copies of these 3087 groups
gap> grps := List(all, x->PcGroupCode(RandomSpecialPcgsCoded(x),Size(x)));;
gap> grps2:= List(grps,x->PcGroupCode(CodePcGroup(x),Size(x)));;
gap> List(grps, CanonicalPcGroup);;time;
219390 # total runtime in ms using our method
gap> 219390/3087.;
71.069 # average runtime per group in ms

gap> ### compare against existing functionality
gap> ids := List(grps2, IdSmallGroup);;time;
105170 # total runtime in ms using IdSmallGroup

gap> iso := List([1..Size(all)],
     i -> IsomorphismGroups(grps2[i],SmallGroup(ids[i])));; time;
21162593 # total runtime in ms (5+ hours) using IsomorphismGroups

gap> isocf := List([1..Size(all)],
     i -> IsomorphismCubefreeGroups(grps2[i], SmallGroup(ids[i])));; time;
113647 # total runtime in ms using IsomorphismCubefreeGroups
```

FIGURE 2. Example computation for solvable groups of size 44100

References

[1] S. Barrera Acevedo, H. Dietrich, M. Horn. The GAP package `CanonicalPcPres`. `gap-packages.github.io/CanonicalPcPres` (to appear soon, 2026).

[2] H. U. Besche, B. Eick. Construction of finite groups. J. Symb. Comput. 27:387-404 (1999), DOI: 10.1006/jsco.1998.0258.

[3] H. U. Besche, B. Eick, E. A. O'Brien. A millennium project: constructing small groups. Int. J. Algebra Comput. 12:623-644 (2002), DOI: 10.1142/S0218196702001115.

```
gap> G:=PcGroupCode(306176090831857578677740122334079584562761156855803601875414432204869061865372949904396907556146355289760477732172359947740788136050202628887574430701817767501856273081248403073820852824978835594488813412792152406292181068000009150915011717294501050327990305,2375789332049525700);
< pc group of size 2375789332049525700 with 18 generators>
gap> ### 2375789332049525700 = 2^2.3^2.5^2.7^2.11^2.17^2.19^2.29.31.47.101
gap> ### derived series section: C66671 x C19, C937750624650 x C2
gap> CanonicalPcGroup(G);;time;
891 # runtime in ms

gap> K := PcGroupCode(6404126914497197298846384046706517423255874685922634397602273938986796593121462613697780016468090688462340534475719485555707030297569169619911971693446011551321290615327993456771170928212982829453285910412729044291276997145293780937934267592306305702698955505927100115900013110033333, 1496880000);
< pc group of size 1496880000 with 18 generators>
gap> ### 1496880000 = 2^7.3^5.5^4.7.11
gap> ### derived series sections: C924 x C6, C270 x C2, C10^2, C5
gap> CanonicalPcGroup(K);;time;
2888 # runtime in ms

gap> L := PcGroupCode(3518419972248533038202112274944177161667028051061861708922400027993796038203852540892781066918474565557823066856068507094625623571536928920276055122571122222019893074110791999221158841053182399587758756292604532025774237230637885169016021139203817376582484458113875162241816773631697220778564941098213080156838310983729151005902727041013277839989390747748298861582018633706626517291024984607000900222219360405072325179361229800428818874603218573908504400675820542129598504065607732189709819535546578020220570283546749879195805467956994922219185651490529614115050520744616295447605980712855245315069988543965361317149349542513019173683 2, 55987200);
< pc group of size 55987200 with 19 generators>
gap> ### 55987200 = 2^10.3^7.5^2
gap> ### derived series sections: C2^2, C3, C10^2, C3^6, C2^6
gap> CanonicalPcGroup(L);;time;
6250 # runtime in ms
```

FIGURE 3. Example computations for three larger groups with derived lengths 2, 4, and 5, respectively

[4] W. Bosma, J. Cannon, C. Playoust. The magma algebra system I: the user language. J. Symb. Comput. 24:235-265 (1997), DOI: 10.1006/jsco.1996.0125.

[5] P. A. Brooksbank, H. Dietrich, J. Maglione, E. A. O'Brien, J. B. Wilson. Categorification of characteristic structures. Forum Math. Sigma 14:e13 (2026), DOI: 10.1017/fms.2025.10159.

[6] J. J. Cannon, B. Eick, C. R. Leedham-Green. Special polycyclic generating sequences for finite soluble groups. J. Symb. Comput. 38:1445-1460 (2004), DOI: 10.1016/j.jsc.2004.05.003.

[7] J. J. Cannon, D. F. Holt. Automorphism group computation and isomorphism testing in finite groups. J. Symb. Comput. 35:241–267 (2003), DOI: 10.1016/S0747-7171(02)00133-5.

[8] H. Dietrich, B. Eick. Coclass graphs of p-groups. Springer Monographs in Mathematics, Springer, 2026, DOI: 10.1007/978-3-032-17506-9.

[9] H. Dietrich, B. Eick, H. Schanze. Group identification of groups of order p^5. Submitted.

[10] H. Dietrich, A. Hulpke. Universal covers of finite groups. J. Algebra 596:681-712 (2021) DOI: 10.1016/j.jalgebra.2020.10.032

[11] H. Dietrich, J. B. Wilson. Isomorphism testing of groups of cube-free order. J. Algebra 545:174-197 (2020), DOI: 10.1016/j.jalgebra.2019.02.008.

[12] H. Dietrich, J. B. Wilson. Group isomorphism is nearly-linear time for most orders. 2021 IEEE 62nd Annual Symposium on Foundations of Computer Science-FOCS 2021, 457-467. IEEE Computer Society, Los Alamitos, CA, 2022, DOI: 10.1109/FOCS52979.2021.00053.

[13] B. Eick, M. Horn. The construction of finite solvable groups revisited. J. Algebra 408:166-182 (2014), DOI: 10.1016/j.jalgebra.2013.09.028.

[14] B. Eick, C. R. Leedham-Green, E. A. O'Brien. Constructing automorphism groups of finite p-groups. Comm. Algebra, 30(5):2271-2295 (2002), DOI: 10.1081/AGB-120003468.

[15] G. Gamble, W. Nickel, E. A. O'Brien. ANUPQ - a refereed GAP package. gap-packages.github.io/anupq.

[16] GAP – Groups, Algorithms, Programming. www.gap-system.org.

[17] D. Holt, B. Eick, E. A. O'Brien. Handbook of Computational Group Theory, Chapman & Hall/CRC, Boca Raton, FL, 2005, DOI: 10.1201/9781420035216.

[18] D. F. Holt. Computing automorphism groups of finite groups. In: Groups and computation, III, 201-208. Ohio State Univ. Math. Res. Inst. Publ., 8 Walter de Gruyter & Co, 2001, DOI: 10.1515/9783110872743.201.

[19] D. J. A. Howden. Computing automorphism groups and isomorphism testing in finite groups. PhD thesis, University of Warwick, 2012, wrap.warwick.ac.uk/50060/.

[20] A. Hulpke. The perfect groups of order up to two million. Math. Comp. 91(334), 1007–1017 (2022), DOI: 10.1090/mcom/3684.

[21] G. Ivanyos, E. J. Mendoza, Y. Qiao, X. Sun, C. Zhang. Faster Isomorphism Testing of p-Groups of Frattini Class 2. 2024 65th Annual IEEE Symposium on Foundations of Computer Science-FOCS 1408-1424 (2024), DOI: 10.1137/24M1719839.

[22] J. Rodney, M. F. Newman, E. A. O'Brien. The groups of order 128. J. Algebra 129:136-158 (1990).

[23] C. Jefferson, R. Waldecker, W. A. Wilson. Computing canonical images in permutation groups with Graph Backtracking. J. Comp. Algebra 8:100010 (2023), DOI: 10.1016/j.jaca.2023.100010.

[24] B. D. McKay, A. Piperno. Practical graph isomorphism, II. J. Symb. Comput. 60:94-112 (2014), DOI: 10.1016/j.jsc.2013.09.003.

[25] J. Müller, M. Neunhöffer, F. Noeske. The GAP package Orb. gap-packages.github.io/orb/.

[26] E. A. O'Brien. Isomorphism testing for p-groups. J. Symb. Comput. 17(2):133-147 (1994), DOI: 10.1006/jsco.1994.1007.

[27] E. A. O'Brien. A guide to the ANU p-Quotient Program, Version 1.8. math.rwth-aachen.de/homes/Greg.Gamble/ANUPQ/standalone-doc/guide.pdf

[28] D. J. S. Robinson. Applications of cohomology to the theory of groups. In: Groups-St. Andrews 1981, 46-80, Cambridge Univ. Press, Cambridge-New York, 1982; London Math. Soc. Lecture Note Ser., 71. Cambridge University Press, Cambridge-New York, 1982, DOI: 10.1017/CBO9780511661884.005.

[29] P. Schweitzer, D. Wiebking. A unifying method for the design of algorithms canonizing combinatorial objects. STOC'19–Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, 1247–1258. Association for Computing Machinery, New York, 2019, DOI: 10.1145/3313276.3316338.

[30] R. Schwingel. Two Matrix Group Algorithms with Applications to Computing the Automorphism Group of a Finite p-Group. PhD thesis. Queen Mary, University of London, 2000, qmro.qmul.ac.uk/xmlui/handle/123456789/67238.

[31] M. J. Smith. Computing automorphisms of finite soluble groups. PhD thesis, Australian National University, 1994, DOI: 10.25911/5d723c5c57106.

[32] D. Wiebking. Normalizers and permutational isomorphisms in simply-exponential time. Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, 230-238. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2020, DOI: 10.1137/1.9781611975994.14.

[33] Y. Qiao, X. Sun. Canonical Forms for Matrix Tuples in Polynomial Time. 2024 IEEE 65th Annual Symposium on Foundations of Computer Science-FOCS 2024, 780-789, Chicago, IL, USA, 2024, DOI: 10.1109/FOCS61266.2024.00054.