RP TU Rheinland-Pfälzische Technische Universität Kaiserslautern Landau

Advanced Algorithms For Induced Sequences And Residual Nilpotence In Polycyclic Groups

Max Mayer

Datum der Disputation:	20. September 2024
erster Gutachter:	Prof. Dr. Max Horn
zweiter Gutachter:	Prof. Dr. Bettina Eick

Vom Fachbereich Mathematik der Rheinland-Pfälzischen Technischen Universität Kaiserslautern-Landau zur Verleihung des akademischen Grades Doktor der Naturwissenschaften (Doctor rerum naturalium, Dr. rer. nat.) genehmigte Dissertation

DE-386

Opus doctorale
lumini meo
dedicatum est.

Zusammenfassung

Polyzyklische Gruppen stellen aus Sicht der Computeralgebra eine interessante Klasse von endlich erzeugten Gruppen dar. Sie sind dadurch charakterisiert, dass sie über eine Normalreihe mit zyklischen Quotienten verfügen. Dadurch lassen sich Elemente polyzyklischer Gruppen im Computer durch ganzzahlige Exponentenvektoren darstellen, was es erlaubt explizite Berechnungen in diesen Gruppen durchzuführen.

Die technischen Entwicklungen in den letzten fünfzig Jahren ermöglichten und förderten das Interesse an komplexeren Fragestellungen in diesem Bereich. Beim Arbeiten insbesondere mit unendlichen polyzyklischen Gruppen stellt sich aber oft das Problem, dass schnelle Berechnungen durch exponentielles Wachstum der involvierten Exponentenvektoren in Zwischenschritten erschwert werden. Insbesondere das Ermitteln induzierter Sequenzen für Untergruppen, ein zentraler Baustein der meisten spezialisierteren Algorithmen, erwies sich in der Vergangenheit als anfällig für derartige Probleme. In der vorliegenden Arbeit entwickeln wir effektive Methoden, die diese Problematik adressieren. Dabei nutzen wir Parallelen der induzierten Sequenzen mit Hermite-Normalformen ganzzahliger Matrizen aus, deren Berechnung bereits über eine breite theoretische Grundlage verfügen, und übertragen diese in den polyzyklischen Kontext.

Im zweiten Teil dieser Arbeit beschäftigen wir uns mit der Frage, wie man eine gegebene polyzyklische Gruppe auf residuelle Nilpotenz untersucht. Eine Gruppe G heißt residuell nilpotent, wenn jedes ihrer nicht-trivialen Elemente ein nicht-triviales Bild in einem nilpotenten Quotienten von G besitzt. Diese Fragestellung ist bereits für einige andere Klassen von Gruppen (bspw. freie Gruppen) untersucht worden. Im polyzyklischen Fall ist bekannt, dass alle polyzyklischen Gruppen einen residuell nilpotenten Normalteiler von endlichem Index besitzen. Es erweist sich aber als deutlich schwerer zu überprüfen, ob bereits die Gruppe selbst residuell nilpotent ist. Bisherige Algorithmen für polyzyklische Gruppen besitzen jedoch nur theoretischen Charakter und wurden nicht zur praktischen Anwendung konzipiert. In dieser Arbeit wollen wir dieses Problem angehen und präsentieren Lösungen für den wichtigen Teilfall der polyzyklischen Gruppen, die einen abelschen Normalteiler N besitzen, sodass der Quotient G/N nilpotent ist. Diese Gruppen treten häufig natürlich als Erweiterungen frei abelscher Gruppen auf (etwa in der Zahlkörpertheorie) und dienen der Konstruktion einer Vielzahl interessanter Beispiele. Dabei ist die Hoffnung, dass die hier gewonnenen Erkenntnisse dabei helfen, algorithmische Lösungen für den allgemeineren polyzyklischen Fall zu entwickeln.

Abstract

Polycyclic groups are from a computer algebraic perspective an interesting class of finitely generated groups. They are characterised by a normal series with cyclic factor groups. One can use this to represent elements of polycyclic groups as integer exponent vectors which allows explicit computations within these groups.

The technological developments within the last fifty years enabled and fostered an increasing interest in more complex questions in this field. When working with infinite polycyclic groups one often faces the problem that fast computations are impeded by an exponential growth of the involved exponent vectors of intermediate results. Especially computing induced sequences for subgroups, which is an important core step for most specialized algorithms, proved to be vulnerable to these issues in the past. This work develops efficient methods which address these issues. For this we utilize the similarities between induced sequences and hermite normal forms for integer matrices, whose computation possess a broad theoretical foundation, which we transfer onto the polycyclic setting.

In the second part of this work we look at the question on how to decide whether a polycyclic group is residually nilpotent. A group G is called residually nilpotent if every non-trivial element has a non-trivial image in a nilpotent quotient of G. This question was already studied for other classes of groups (e.g. free groups). It is known that all polycyclic groups possess a residually nilpotent normal subgroup of finite index. However, deciding whether the original group is residually nilpotent itself appears to be more challenging. Existing Algorithms for polycyclic groups only possess theoretical character and haven't been designed for actual usage. In this work we want to address this issue and present effective solutions for the important subcase of polycyclic groups which have an abelian normal subgroup N such that the quotient group G/N is nilpotent. These groups often arise naturally as extensions of free abelian groups e.g. in number field theory and are used to construct many interesting examples. Understanding these groups might also help in developing a more general theory for arbitrary polycyclic groups.

Contents

Preface

In 1938 Hirsch published his article "On infinite soluble groups" [Hir38]. This article together with its four successors laid the groundwork to the study of what Hirsch called S-groups and are known as polycyclic groups today. A group G is called polycyclic if it has a series $G = G_1 \trianglerighteq G_2 \trianglerighteq \ldots \trianglerighteq G_n \trianglerighteq 1$ where each quotient G_i/G_{i+1} is cyclic. By choosing generators of these quotients one can represent group elements as integer exponent vectors. Hirsch showed that the number of infinite cyclic quotients in such a series is an invariant and polycyclic groups satisfy the maximality condition for subgroups, i.e. every subgroup of a polycyclic group is finitely generated. The class of polycyclic groups is nested between the solvable and the finitely generated nilpotent groups. This sparked interest in the further research of this class of groups and a wide array of properties have been shown since.

For instance, Hirsch showed that the Fitting subgroup of a polycyclic group is nilpotent [Hir54], which was also proven independently by Itô [Itô53]. Later it was shown in [BCR91] that this subgroup, along with multiple other properties of polycyclic groups, can be determined algorithmically. However, the early algorithms presented there as well as those in [Seg90] had purely theoretical character and weren't designed for computational use. By a theorem of Mal'cev [Mal51] every solvable linear group over the integers is polycyclic. Conversely, Auslander [Aus67] and Swan [Swa67] proved that all polycyclic groups can faithfully be represented as unimodular matrix groups. This meant that algorithms for matrix groups could directly be applied to polycyclic groups and for further research on this we refer the reader to [Luk92] and [Bea97]. One of the first accounts on practical algorithms specifically for polycyclic groups was given in [Sim94]. In [Eic00] the author developed many fundamental algorithms applicable to arbitrary polycyclic groups which laid the foundation to many modern algorithmic developments in this field. These algorithms have since been implemented as a package [POLY] for the computer algebra system GAP [GAP] in the early 2000s.

When working with infinite polycyclic groups one often faces the problem that

fast computations are impeded by an exponential growth of the involved exponent vectors of intermediate results. In fact, a core step in many algorithms for polycyclic groups, namely computing induced sequences for subgroups, proved to be very vulnerable to these issues in the past. An induced sequence is a set of generators, which define a polycyclic series of the subgroup generated by it that refines the series of the parent group. Induced sequences and the algorithm to compute them bear a striking resemblance to hermite normal forms for integer matrices. In [Eic21] the author presented an updated version to this algorithm but left it open to resolve the issue of coefficient explosion. Addressing this is one of the main goals of this thesis. For this we aim to utilize the well-developed infrastructure on hermite normal form computation and apply them to the polycyclic setting by partitioning the polycyclic series into abelian quotients. We cover this in the first half of this work.

In the second part we study the problem on how to decide whether a polycyclic group is residually nilpotent. A group G is called residually $\mathcal{X}$, where $\mathcal{X}$ is a property of groups, if every non-trivial element has a non-trivial image in an $\mathcal{X}$ quotient of G. The question has been answered for some classes of groups, e.g. Magnus showed that free groups are residually (torsion-free) nilpotent [Mag35]. Partial results are available for one-relator groups (see [Bau68] and [McC96]). For polycyclic groups it was shown by Mal'cev [Mal40] that they contain a residual nilpotent subgroup of finite index. Semidirect products and group extensions of residually nilpotent groups are studied in [FR85] and more recently in [BNB21]. In this work we are developing algorithmic solutions for polycyclic groups G which have an abelian normal subgroup N such that the quotient is nilpotent. These groups often arise naturally as extensions of free abelian groups e.g. in number field theory and are used to construct many interesting examples. For this purpose, we are using a criterion given in [Rob82] to translate the problem into an eigenvalue problem of a unimodular matrix group and introduce techniques to resolve it. Understanding this subcase might also help in developing a more general theory for arbitrary polycyclic groups.

The thesis is structured in the following way: Chapter 1 is an introductory chapter where we formally introduce the notion of polycyclic groups, sequences and presentations. We state a few properties of polycyclic groups which we need throughout this work. For a more detailed account on polycyclic groups we refer the reader to the books by Segal [Seg83], Wehrfritz [Weh09] and Robinson [Rob82]. Chapter 2 contains the discussion about induced sequences. We give a description of the general outline of the algorithm and address each part of the algorithm individually before piecing everything together. In Chapter 3 we describe an application of the new induced sequence algorithm by translating the Zassenhaus algorithm for computing intersections from linear algebra to the polycyclic setting, and compare it to the algorithm described in [Eic00]. The first half of chapter 4 is taken up by a more general discussion of residual

nilpotence in polycyclic abelian-by-nilpotent groups and the theoretical background necessary for developing the algorithms presented in its second half. We finish this thesis with chapter 5 where we give a brief summary and point out few open problems and possible directions on how to proceed from here.

1. Preliminaries

1.1. Definition
A group G is called *polycyclic* if it has a finite subnormal series with cyclic factors. I.e. a series $G = G_1 \trianglerighteq G_2 \trianglerighteq \cdots \trianglerighteq G_n \trianglerighteq \{1\}$ where G_i/G_{i+1} is cyclic. We call the subnormal series a *polycyclic series*.
Let $g_1, \ldots, g_n$ be elements of the group G. If the subgroups $G_i := \langle g_i, \ldots, g_n \rangle$ form a polycyclic series, we say that these g_i form a *polycyclic sequence*.

For every polycyclic series one can construct a polycyclic sequence by choosing generators for the quotients G_i/G_{i+1}. As this choice is not unique neither is the sequence defined by it. In order to keep the notation throughout this work simple we will allow ourselves to just talk about polycyclic sequences $g_1, \ldots, g_n$ and implicitly fix the corresponding series of the G_i.

1.2. Definition
Let G denote a group. A *polycyclic presentation* is a finite set of generators $g_1, \ldots, g_n$ of G and some exponents $e_1, \ldots, e_n \in \mathbb{N}_{>0} \cup \{\infty\}$ with relations of the form

$$\begin{aligned} g_j^{g_i} := g_i^{-1} g_j g_i &= g_{i+1}^{x_{i+1,(i,j)}} \cdot \ldots \cdot g_n^{x_{n,(i,j)}} \quad \text{for } 1 \leq i < j \leq n \\ g_j^{g_i^{-1}} := g_i g_j g_i^{-1} &= g_{i+1}^{y_{i+1,(i,j)}} \cdot \ldots \cdot g_n^{y_{n,(i,j)}} \quad \text{for } 1 \leq i < j \leq n \\ g_j^{e_j} &= g_{j+1}^{z_{j,j+1}} \cdot \ldots \cdot g_n^{z_{j,n}} \quad \text{for } e_j \neq \infty \end{aligned}$$

where $x_{k,(i,j)}, y_{k,(i,j)}$ and $z_{l,k}$ are integers for all k with $e_k = \infty$ or natural numbers less than e_k if the latter is finite.

The relations of a polycyclic presentation imply that the $g_1, \ldots, g_n$ form a polycyclic sequence and G is a polycyclic group. Conversely, any polycyclic sequence can be used to define a polycyclic presentation [HEO05, thm. 8.8].
An important property of polycyclic groups is that they satisfy the maximality condition:

1.3. Proposition
Let G be a polycyclic group. Then every ascending chain of subgroups of G becomes stationary. In particular every subgroup of a polycyclic group is finitely generated.

Proof. See [Seg83, ch. 1.A prop. 4]. □

1.4. Definition
Let $G = G_1 \trianglerighteq \ldots \trianglerighteq G_{n+1} = 1$ be a polycyclic series. We call the index $\mathrm{rord}_i(G) := [G_i : G_{i+1}]$ the i-th *relative order* and write $\mathrm{rord}(G) = (\mathrm{rord}_1(G), \ldots, \mathrm{rord}_n(G))$. The *Hirsch number* $\mathrm{h}(G)$ (or *Hirsch length*) is the number of relative orders which are infinite and is an invariant of the group G [Rob82, 5.4.13]. If all relative orders are infinite, we say that G is *poly-C_∞*.

1.5. Definition
Let G be a polycyclic group given by a polycyclic presentation with generators $g_1, \ldots, g_n$ and exponents $e_1, \ldots, e_n \in \mathbb{N}_{>0} \cup \{\infty\}$. Then for an element g of G there are integers $x_1, \ldots, x_n$ where $0 \leq x_i < e_i$ for all i with $e_i < \infty$ such that $g = g_1^{x_1} \cdot \ldots \cdot g_n^{x_n}$. If these x_i are unique, we call $\exp(g) := (x_1 \ \ldots \ x_n) \in \mathbb{Z}^n$ the *exponent vector* of g.

The proof of the existence of such an exponent vector can be found in [Eic00].
Explicitly computing $\exp(g)$ for an element $g \in G$ can be done by writing g as a product of the g_i and whenever one finds an unordered pair $g_j \cdot g_i$ where $i < j$ use the conjugate relations in 1.2 to replace it by an ordered expression (similar use of the exponent relations). This procedure is called *collection* and its result depends on the order in which the unordered subwords are replaced. However, one can show that for every polycyclic group there is a polycyclic presentation such that the outcome is unique. Such a presentation is typically called *consistent* or *confluent* and there are means to check this property. For more details see [Eic00, section 2.3]. For our purposes we simply assume that whenever a polycyclic presentation is given it is also a consistent one. This also justifies talking about *the* exponent vector of an element and allows us to compare elements of a polycyclic group simply by comparing the respective exponent vectors.
Another important thing to keep in mind when considering collection is that its runtime also depends on the order in which it is executed. There are different strategies and algorithms for special classes of polycyclic groups to improve the performance which can be considered as an entire field of research on its own. For instance, [Hal69, thm. 6.5] showed that in the case of finitely generated torsion-free and nilpotent groups there exist certain polynomials which can be evaluated to perform symbolic collection. Algorithms to compute these polynomials are described in [LGS98] and [Sim94, ch. 9.10]. However, for

general polycyclic groups there is a limit to this as the following example illustrates.

1.6. Example
We look at the group

$$\begin{aligned} G = \langle g_1, g_2, g_3 \mid & g_2^{g_1} = g_2^{-2} g_3^5,\ g_2^{g_1^{-1}} = g_2^7 g_3^5, \\ & g_3^{g_1} = g_2^3 g_3^{-7},\ g_3^{g_1^{-1}} = g_2^3 g_3^2, \\ & g_3^{g_2^{\pm 1}} = g_3 \rangle. \end{aligned}$$

This presentation is consistent so the exponent vector of an element of the group is unique. In this group the following holds:

$$\begin{aligned} g_3 g_1 &= g_1 g_2^3 g_3^{-7} \\ g_3 g_1^2 &= g_1 g_2^{-27} g_3^{64} \\ g_3 g_1^3 &= g_1 g_2^{246} g_3^{-583} \\ g_3 g_1^4 &= g_1 g_2^{-2241} g_3^{5311} \\ &\vdots \\ g_3 g_1^{10} &= g_1 g_2^{-1280816685} g_3^{3035438299}. \end{aligned}$$

So despite all numbers involved in the presentation being rather small (with absolute value less than 10) the length (i.e. the number of digits) of the exponent vector of $g_3 g_1^n$ already grows exponentially in the length of n. Therefore, if already just writing down the result itself requires exponential space, there is no hope for a fast (in the sense of polynomial time) collection algorithm.

Hence, the rule of thumb here is the less we need to use collection in our algorithms the better the overall performance will be.
We fix some additional notation.

1.7. Definition
Let G be a polycyclic group given by a polycyclic sequence $g_1, \ldots, g_n$ and $g \in G$ with exponent vector $(e_1 \ \ldots \ e_n)$ with respect to the g_i. We define the *depth* of g as

$$\operatorname{depth}(g) = \begin{cases} \min\{i \in \{1, \ldots, n\} \mid e_i \neq 0\}, & \text{if } g \neq 1 \\ n+1, & \text{else.} \end{cases}$$

The *lead* of that element is then defined as

$$\operatorname{lead}(g) = \begin{cases} e_{\operatorname{depth}(g)}, & \text{if } g \neq 1 \\ 1, & \text{else.} \end{cases}$$

Note that this definition depends on the provided sequence.

1.8. Remark
An important thing to notice is how lead and depth interact with the group multiplication. If G is a polycyclic group and $g,\ h \in G$ then $\mathrm{depth}(g \cdot h) \geq \min(\mathrm{depth}(g), \mathrm{depth}(h))$ where equality holds if $\mathrm{depth}(g) \neq \mathrm{depth}(h)$. This follows directly from the structure of the relations in a polycyclic presentation as during collection the component of smallest depth just propagates through. If $\mathrm{depth}(g) \neq \mathrm{depth}(h)$, the lead of the product will therefore be the lead of the factor with smallest depth. In the case where both g and h are of the same depth d, the lead of both will add up during collection which then might cancel out (as the sum might be zero or because of an exponent relation).

We summarize this observation as

1.9. Lemma
Let G be a polycyclic group given by some polycyclic sequence $G = G_1 \trianglerighteq G_2 \trianglerighteq \cdots \trianglerighteq G_{n+1} = 1$. Let $u, x_1, \ldots, x_k$ be elements of G such that $d := \mathrm{depth}(x_1) < \mathrm{depth}(x_i)$ for $i = 2, \ldots, k$. Denote by e_d the d-th component of the exponent vector of u and write r_d for $\mathrm{rord}_d(G)$ if the index is finite or otherwise set $r_d = 0$. Then if u is an element of $\langle x_1, \ldots, x_k \rangle$ we have

$$e_d \in \langle \mathrm{lead}(x_1), r_d \rangle_{\mathbb{Z}}.$$

The following definition will be extensively used in chapter 4.

1.10. Definition
Let G be a group and $\mathcal{A}$ and $\mathcal{B}$ properties of groups (e. g. finite, abelian, etc.). Then G is called an $\mathcal{A}$-*by*-$\mathcal{B}$ group if it has a normal subgroup N which has property $\mathcal{A}$ and the quotient G/N has property $\mathcal{B}$.

Note that one needs to be careful as some authors might swap the meaning of $\mathcal{A}$ and $\mathcal{B}$ in the above definition.

1.11. Remark
If $\mathcal{A}$ and $\mathcal{B}$ are properties inherited by subgroups, then also subgroups of an $\mathcal{A}$-by-$\mathcal{B}$ group are themselves $\mathcal{A}$-by-$\mathcal{B}$. Analogously, if $\mathcal{A}$ and $\mathcal{B}$ are properties inherited by quotients, then also quotients of an $\mathcal{A}$-by-$\mathcal{B}$ group are $\mathcal{A}$-by-$\mathcal{B}$. Note that a polycyclic-by-polycyclic group is polycyclic itself. [Sim94, ch. 9 prop. 3.3]

2. Fast Induced Sequence Computation

This chapter describes a new fast algorithm for computing induced sequences for subgroups of infinite polycyclic groups. In section 1 of this chapter, we introduce the notion of induced sequences together with some relevant properties and notation regarding the uniqueness of such a sequence. For more details, the reader should refer to [HEO05] and [Eic00]. In section 2 we state the general outline of the algorithm to compute such a sequence. We follow this up with a short discussion on the issues previous induced sequence algorithms had, which we aim to solve with our approach. Section 3 addresses the core procedure of the algorithm, i.e. given a set of generators of a subgroup to compute a new (usually) smaller set of generators of disjoint depths, which we will call an echelon form, that satisfy a certain inclusion property. Previous algorithms were only feasible for smaller examples as they are vulnerable towards an explosion of the average coefficient size and exponential growth in the number of elements computed in intermediate steps. To resolve these issues we introduce the notion of abelian segmentations which allow us to make direct use of the advanced algorithms developed for computing Hermite normal forms of integer matrices. In section 4 we state a few technical results which we need to show termination of the induced sequence algorithm. The discussion about this is necessary as previous termination criteria are incompatible with our approach. We put everything together in section 5 where we state the complete algorithm 2.33 together with its proof. In section 6 we add a discussion on how to determine "good" abelian segmentations for a polycyclic group in the sense that they allow for an efficient use of the induced sequence algorithm. For this purpose we state two greedy algorithms working "top-down" and "bottom-up" which produce minimal sized segmentations. Afterwards we compare both approaches using our GAP implementation of algorithm 2.33. Finally, we conclude this chapter with a few runtime tests and compare our new algorithm with previous implementations.

Note that we simplify the notation in this chapter: Unless stated otherwise, we will assume whenever a polycyclic group is given by a polycyclic series $G = G_1 \trianglerighteq G_2 \trianglerighteq \cdots \trianglerighteq G_n \trianglerighteq \{1\}$ we also fix some sequence generating the G_i. Furthermore, we assume that the inclusions of the G_i are strict to avoid special cases where the quotient is trivial. In particular, the definition of *depth* and *lead* should refer to that sequence. This means that if we have some $u \in U \leq G$ the expression $\mathrm{depth}(u)$ should refer to its depth in G rather than to the one in U.

2.1. Induced Sequences

2.1. Definition
Let G be a polycyclic group given by a polycyclic sequence and U a subgroup. Let $G = G_1 \triangleright G_2 \triangleright \cdots \triangleright G_n \triangleright \{1\}$ be a polycyclic series of G and let $u_1, \ldots, u_k \in G$ define a polycyclic sequence of $U = \langle u_1, \ldots, u_k \rangle$. We call this sequence an *induced (polycyclic) sequence* if it is strictly descending and consists of all intersections $U \cap G_i$.

2.2. Lemma
Let G be a polycyclic group and $U \subset G$ a subgroup. If $g \in G$ is an element such that $g^{-1}Ug \subseteq U$ holds, then we also have $gUg^{-1} \subseteq U$.

Proof. Follows directly from [Sim94, ch.9 prop. 3.12]. □

2.3. Corollary ([HEO05, lemma 8.34])
Let $G = G_1 \triangleright \cdots \triangleright G_n \triangleright \{1\}$ be a polycyclic group and let $u_1, \ldots, u_k \in G$ define the subgroups $U_i = \langle u_i, \ldots, u_k \rangle$. Then $U_1 \geq U_2 \geq \cdots \geq U_k$ is an induced polycyclic sequence for $U = U_1$ if and only if

1. $u_j^{u_i} \in U_{i+1}$ for all $1 \leq i < j \leq k$,
2. $u_i^{a_i} \in U_{i+1}$ if $a_i := [\langle u_i, G_{d+1} \rangle : G_{d+1}]$, where $d = \mathrm{depth}(u_i)$, is finite, and
3. $\mathrm{depth}(u_i) < \mathrm{depth}(u_j)$ for $i < j$.

2.4. Remark
Note that condition 1 of corollary 2.3 can be replaced by

$$[u_j, u_i] = u_j^{-1} u_i^{-1} u_j u_i \in U_{i+1} \text{ for all } 1 \leq i < j \leq k$$

as $u_j \in U_{i+1}$ for all $j > i$. Which one to prefer depends on the overall context. We will use this in some cases as it is easier to talk about commutators in

some proofs rather than conjugates. From a computational view, computing the commutator requires an additional multiplication and inversion compared to just computing the conjugate. On the other hand, depending on the given group the commutator might have a higher depth which reduces the complexity of the following computations. So which one to prefer often depends on the explicit example.

When working with $\mathbb{Z}$-modules and their submodules the notion of an *Hermite normal form* of an integer matrix is often used to decide a variety of problems. The algorithms presented within this work share a lot of similarities with the computation of such an Hermite normal form. However, the definition of an Hermite normal form is not standardized and varies in the literature. That's why we state a definition here that is most suitable for our purposes.

2.5. Definition
Let $A = (a_{i,j})_{i,j} \in \mathrm{Mat}(m \times n, \mathbb{Z})$ be a matrix. We say that A is in *(row) echelon form* if there is an r and integers $1 \leq j_1 < \ldots < j_r \leq n$ such that we have

1. $a_{i,j_i} \neq 0$ for all $i = 1, \ldots, r$,
2. $a_{i,k} = 0$ for all $i = 1, \ldots, r$ and $k = 1, \ldots, j_i - 1$ and
3. $a_{l,k} = 0$ for all $l > r$ and $k = 1, \ldots, n$.

The j_i are called the *pivots* of A.
We say that A is in *Hermite normal form* if it is in echelon form and also for any non-zero row i

1. the entry a_{i,j_i} is positive and
2. $0 \leq a_{l,j_i} < a_{i,j_i}$ for all $l < i$.

For every matrix $A \in \mathrm{Mat}(m \times n, \mathbb{Z})$ there is a matrix H in Hermite normal form with the same row space as A and one can compute a transformation matrix $T \in \mathrm{GL}_m(\mathbb{Z})$ with $H = TA$ (see for instance [Coh93, 2.4.4]). Note that the definition for the Hermite normal form given here can be easily replaced by any other commonly used one. The algorithms presented in this work only need to be adapted slightly.
The following notation makes things easier to understand later:

2.6. Remark
Let G be a polycyclic group given by a polycyclic presentation with n generators and let $m \in \mathbb{N}$. By associating row vectors in $\mathbb{Z}^n$ with elements in G we can define a map to the power set of G

$$\mathrm{Mat}(m \times n, \mathbb{Z}) \to \mathcal{P}(G),\ M \mapsto \{g \in G \mid \exp(g) \text{ is a row of } M\}.$$

We say that a subgroup $U \leq G$ *is generated by* $M \in \mathrm{Mat}(m \times n, \mathbb{Z})$ if it is generated by the image of M under this map. Conversely, a finite set of m elements in G can be associated to a matrix in $\mathrm{Mat}(m \times n, \mathbb{Z})$ by writing the exponent vectors as the rows. By fixing an ordering on the set $\mathbb{Z}^n$ this matrix can be made unique. Note that condition 3 of lemma 2.3 translates to requiring the set of generators of an induced sequence to be associated with a matrix in row echelon form. We therefore say that a set of elements $u_1, \ldots, u_k$ of a polycyclic group is in *echelon form* or respectively in *Hermite normal form* if the corresponding matrix (with respect to the ordering) is. This is why the computation of an induced polycyclic sequence is often viewed as a generalization of the computation of an Hermite normal form of an integer matrix. We define the following *elementary row operations*:

- *Swapping row i and j.*
- *Multiplying a row by an integer $z \in \mathbb{Z}$:* The z-multiple of a row is defined as the exponent vector of the z-th power of the element corresponding to this row.
- *Adding the z-multiple of the row i to the row $j \neq i$ from the left (from the right)* replaces the row j by the exponent vector of the element given by row i multiplied from the left (the right) by the z-th power of the element given by row j.

If G is a free abelian group this definition coincides with the usual definition from linear algebra where one multiplies a row vector by an integer z simply by multiplying each component of the vector. Notice that neither swapping rows nor adding a multiple of a row to another changes the subgroup which is generated by a matrix. However, multiplying a row by an integer z usually does. In this case we want to enlarge the matrix by the new row vector instead to ensure that the generated subgroup is still the same.

While it is clear that the series obtained of an induced polycyclic sequence $u_1, \ldots, u_k$ of a subgroup is unique by definition, the sequence itself is not. However, it is possible to enforce uniqueness by requiring additional properties which is the purpose of the following definitions. Choosing a canonical set of generators is usually convenient as it allows us to check whether two given subgroups are equal simply by comparing their canonical generators.

2.7. Definition
Let G be a polycyclic group given by a polycyclic presentation and $g, h \in G$ of depth d. We call h a *normalization* of g if $h = g^e$ for some $e \in \mathbb{Z}$ and

- $\mathrm{lead}(h) = |\mathrm{lead}(g)|$ if $G_d/G_{d+1} \simeq \mathbb{Z}$ or

- $\mathrm{lead}(h) = |\gcd(|G_d/G_{d+1}|, \mathrm{lead}(g))|$ if $\mathrm{rord}_i(G)$ is finite.

We call g *normalized* if g is a normalization of itself.

2.8. Remark
If we look at the group generated by the image of g in the cyclic group G_d/G_{d+1}, it becomes clear that the definition of a normalization is chosen in such a way that any such h has the unique smallest positive generator as lead exponent. Note that h is not unique. Only its class in G_d/G_{d+1} is.

Computing a normalization is simple:

Algorithm 2.9: Normalization

Input: an element g of a polycyclic group G of $\mathrm{depth}(g) = d$
Output: a normalization of g

1 **if** $\mathrm{rord}_d(G) = \infty$ **then**
2 Return $g^{\mathrm{sgn}(\mathrm{lead}(g))}$.
3 **else**
4 Compute $a, b \in \mathbb{Z}$ such that

$$\gcd(\mathrm{lead}(g), \mathrm{rord}_d(G)) = a \cdot \mathrm{lead}(g) + b \cdot \mathrm{rord}_d(G)$$

5 Return g^a.

2.10. Lemma
Let G be a polycyclic group and $g \in G$ a normalized element. Then any element $h \in G$ with $\mathrm{depth}(h) = \mathrm{depth}(g)$ and $\mathrm{lead}(h) > 0$ and dividing $\mathrm{lead}(g)$ is normalized as well.

Proof. If $G_d/G_{d+1} \simeq \mathbb{Z}$ this is immediate by $\mathrm{lead}(h) > 0$. So assume that $\mathrm{rord}_d(G) < \infty$ and we have

$$\mathrm{lead}(g) = |\gcd(\mathrm{rord}_d(G), \mathrm{lead}(g))|.$$

Hence, $\mathrm{lead}(h) > 0$ dividing $\mathrm{lead}(g)$ implies $\mathrm{lead}(h) = |\gcd(\mathrm{rord}_d(G), \mathrm{lead}(h))|$. □

2.11. Definition
Let G be a polycyclic group given by a polycyclic presentation and $u_1, \ldots, u_k \in G$ with $\mathrm{depth}(u_i) < \mathrm{depth}(u_j)$ for all $i < j$. Writing $d_i = \mathrm{depth}(u_i)$ and $e_i = \exp(u_i)$ for all $i = 1, \ldots, k$, we call $u_1, \ldots, u_k$ *reduced* if for all $1 \le i < j \le k$ we have

$$\mathrm{lead}(u_j) > (e_i)_{d_j} \ge 0.$$

Algorithm 2.12: Reduction

Input: a set of elements $u_1, \ldots, u_k$ of a polycyclic presented group G with $\text{depth}(u_i) < \text{depth}(u_j)$ for all $i < j$

Output: reduced elements $u'_1, \ldots, u'_k \in G$ generating the same subgroup as $u_1, \ldots, u_k$

```
for i from 1 to k − 1 do
    for j from i + 1 to k do
        Denote by e the depth(u_j)-th component of exp(u_i).
        Use division with remainder to compute
          q, r ∈ ℤ, lead(u_j) > r ≥ 0 such that e = q · lead(u_j) + r.
        Replace u_i by u_i · u_j^{-q}
```

The correctness of algorithm 2.12 follows directly from definition.

2.13. Definition
Let G be a polycyclic group and $U \leq G$. Assume the elements $u_1, \ldots, u_k \in G$ define an induced polycyclic sequence of U. Then we say the sequence is *canonical* if each element u_i is normalized and $\{u_1, \ldots, u_k\}$ is reduced.

The canonical sequence is unique for each subgroup U. For a proof of the uniqueness see for example [Eic00, 3.5].

2.2. Computing An Induced Sequence

The main goal of this section is to give a new efficient algorithm which given a polycyclic series

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

of a group G and a set of generators $u_1, \ldots, u_m$ of a subgroup $U \leq G$ outputs a set of generators $u'_1, \ldots, u'_k$ which define an induced polycyclic sequence of U. Algorithms for this purpose are described in [HEO05, 8.3.1] for finite groups and in [Eic00, 3.1] for arbitrary polycyclic groups. Implementations of these algorithms are provided for the computer algebra system GAP [GAP] in the package Polycyclic [POLY]. However, these algorithms suffer from rather poor performance especially in cases where the number of generators m or the length of the polycyclic sequence n is large. This is due to an exponential blow-up in used memory space at different steps during the computation. In [Eic21] the author of the article revisited the problem but without significantly improving the performance. The goal of this chapter is to give a substantially improved

version of this algorithm. In order to describe the respective issues and how we are going to resolve them, we first give a simplified description of the previous instances of the algorithm:

Algorithm 2.14: Computing induced sequence (general)

Input: a finite set of elements M of a polycyclic group G given by a polycyclic series $G = G_1 \triangleright \ldots \triangleright G_n \triangleright G_{n+1} = 1$

Output: an induced sequence for the subgroup $U = \langle M \rangle$

1 Initialize X as an empty list.
2 Set up a list L with the elements in M.
3 **while** *some halting condition is not met* **do**
4 Choose an element g in L and remove it from L.
5 Compute $Y \subseteq \langle M \rangle$ in echelon form such that for all $1 \leq d \leq n$:

$$\langle u \in X \cup \{g\} \mid \mathrm{depth}(u) \geq d \rangle \leq \langle u \in Y \mid \mathrm{depth}(u) \geq d \rangle$$

6 Replace X by Y.
7 Determine candidates of elements in $X \cap G_d$ which are potentially not in $\langle u \in X \mid \mathrm{depth}(u) \geq d \rangle$ and insert them into L.
8 Return X.

The computational core of this algorithm lies in step 5 when an echelon form is computed from the previous intermediate result and the newly added element g. We address this separately in section 2.3. From this intermediate result (called a *partial induced sequence* [HEO05] or *close sequence* [Eic00]) it depends which elements need to be added in step 7. Note that adding the word *potentially* was intentional as this correlates with the halting condition: A typical condition often used is to check whether L is empty, but this requires some knowledge on when it is unnecessary to add certain elements to L or else the loop would not halt. As we aim to modify the central part of the algorithm, we also have to adapt the halting condition. We will develop the necessary theory in section 2.4 which we will then apply in the termination proof of our final algorithm 2.33.

At least the type of elements added in step 7 is mostly fixed: Corollary 2.3 showed that a set of generators in echelon form define an induced polycyclic sequence if the sequence is closed under taking conjugates (or commutators) and taking powers for elements with finite relative order. So the idea is whenever this is not the case to add the missing conjugate or power to the list of generators and compute a new sequence of generators which refines the previous one.

2.15. Example
There is some leeway when deciding which conjugates or commutators should be added. Using lemma 2.2 we have that if $\langle u_j, \ldots, u_k \rangle$ is closed under con-

jugation with u_i, it is also closed under conjugation with u_i^{-1}. So it suffices to only look at conjugates of the form $u_j^{u_i}$. This reduces the total number of elements during an echelon computation and therefore typically this reduces the overall work in a single computation but there are examples in which also adding $u_j^{u_i^{-1}}$ is more efficient:

Let $\sigma = (1\ 2 \dots n) \in S_n$ be a permutation on n elements and let G be the polycyclic group given by

$$\langle g_0,\ g_1, \dots, g_n \mid g_i^{g_0} = g_{\sigma(i)},\ g_i^{g_0^{-1}} = g_{\sigma^{-1}(i)} \text{ for all } i = 1, \dots, n \text{ and} \\ g_i g_j = g_j g_i \text{ for all } 1 \le i, j \le n \rangle$$

We want to compute an induced generating sequence of $U = \langle g_0, g_1 \rangle$. Let's say we want to add conjugates of the form $u_j^{u_i}$. Then we will add in a first iteration the element $g_1^{g_0} = g_2$, the second iteration only yields $g_2^{g_0} = g_3$ as a new generator. Continuing like this we will end after $n-1$ iterations with the result $U = \langle g_0, \dots, g_n \rangle = G$.

If we had also added the conjugate with the inverse in each iteration we would get the result already after $\lceil \frac{n-1}{2} \rceil$ iterations but we would have had to handle twice as many elements in each echelon form computation. Also as we are going to point out in remark 2.28 that adding more elements then necessary can even have a positive effect on the performance. This is due to certain changes we introduce to the algorithm 2.27 computing an echelon form.

A different approach for the choice of added elements is given by the following lemma. In some cases one might already have an induced sequence of the derived subgroup U' of the subgroup U. In these cases it is already enough to add this sequence to the set of generators of U and compute an echelon form. This approach has the advantage of also giving a bound on how often one needs to add generators. However, in most circumstances it is not applicable as one typically uses an induced sequence for U to compute one for U'. For more details see [Eic00, section 8.1].

2.16. Lemma
Let G be a polycyclic group given by a polycyclic sequence where each quotient is infinite. Let U be a subgroup given by a set of generators $u_1, \dots, u_k$ in echelon form. If there is an induced polycyclic sequence $v_1, \dots, v_l$ of the commutator subgroup U' such that

$$v_j \in \langle u_i \mid \text{depth}(u_i) \ge \text{depth}(v_j) \rangle$$

then $u_1, \dots, u_k$ is an induced generating sequence of U.

Proof. By corollary 2.3 we only need to show that $u_j^{u_i} \in U_{i+1}$ for all $i < j$ or equivalently by 2.4 that $[u_i, u_j] \in U_{i+1}$ as the other two conditions of the

corollary are satisfied by assumption. As this is an element of the commutator subgroup there are integers $e_1, \ldots, e_l$ such that

$$[u_i, u_j] = v_1^{e_1} \cdot \ldots \cdot v_l^{e_l}$$

where $e_m = 0$ if $\text{depth}(v_m) \leq \text{depth}(u_i)$ as the $v_1, \ldots, v_l$ define an induced polycyclic sequence and $\text{depth}([u_i, u_j]) > i$. By our assumption this implies

$$[u_i, u_j] \in \langle v_m \mid \text{depth}(v_m) > \text{depth}(u_i) \rangle \leq \langle u_{i+1} \ldots, u_k \rangle = U_{i+1}. \qquad \square$$

Before we start to discuss how we compute an echelon form for a set of generators we should address the order of operations: Algorithm 2.14 was stated as it is commonly done in previous instances of this algorithm (see [Eic00], [HEO05], [Eic21]). Every time a new element is added and an echelon form is computed one adds the new candidates of generators to the list L. We argue that this is rather inefficient for mostly two reasons and one should rather clear the list L first and afterwards compute a new set of candidates:

The first reason is that adding elements in between keeps the list L unbounded as every time an element is removed form L there are usually multiple new ones added to L. Handling an unbounded list is inefficient as it may require additional memory allocation during the algorithm which reduces the performance. By clearing the list L first and afterwards computing new candidates the list L can be bounded: Starting with m generators for a subgroup U one first computes a set of generators in echelon form. This first computation bounds the set of generators by the number of generators n of the parent group. Afterwards one adds commutators (or conjugates) as well as powers of the generators in the case of finite relative order. Their number in each iteration can therefore be bounded by a constant depending on n.

The second reason why one should avoid constantly adding new candidates before clearing the list L first is that they might become obsolete later. Example 2.17 illustrates this problem.

2.17. Example

We use the same group G as in example 2.15 and fix some $k \in \mathbb{N}$. Assume we want to compute an induced generating sequence of

$$U = \langle g_0,\ g_1^{2^k},\ g_1^{2^{k-1}}, \ldots, g_1 \rangle$$

using algorithm 2.14. It is clear that this is just G as argued before. We assume that the elements are taken from the list L following a first-in-first-out principle and in each iteration only conjugates of the form $u_i^{-1} u_j u_i$ for $i < j$ are added if they are not yet present in the intermediate result X. The table below lists the state of the variables after each iteration. The induced sequence algorithm terminates after $n \cdot k$ steps instead of the minimal $n + k$ steps we would get

iteration	X	L
1	g_0	$g_1^{2^k}, \ldots, g_1$
2	$g_0,\ g_1^{2^k}$	$g_1^{2^{k-1}}, \ldots, g_1, g_2^{2^k}$
$\vdots$	$\vdots$	$\vdots$
k	$g_0,\ g_1$	$g_2^{2^k}, \ldots, g_2$
$k+1$	$g_0,\ g_1,\ g_2^{2^k}$	$g_2^{2^{k-1}}, \ldots, g_2, g_3^{2^k}$
$\vdots$	$\vdots$	$\vdots$
$k \cdot n$	$g_0, \ldots, g_n$	$\emptyset$

if we had just cleared L first before adding conjugates. The same could be achieved by reordering the elements in L first. But deciding which element should be added first would require additional information or might not even be decidable a priori in a less trivial example.

Note that in this simple example the used memory space of L stays bounded as only one conjugate needs to be added in each iteration. In a more general example more conjugates are necessary and the length of L can grow exponentially.

2.3. Computation Of An Echelon Form

As discussed in remark 2.6 computing an echelon form of a set of elements of a polycyclic group has a lot in common with Hermite normal form computations for integer matrices. In fact one can directly adapt the basic textbook algorithm (see for instance [Coh93, 2.4.4]) to the polycyclic setting:

Algorithm 2.18: Echelon form

Input: a non-empty set of elements $U := \{u_1, \ldots, u_m\}$ of a polycyclic group G given by a polycyclic presentation on n generators $g_1, \ldots, g_n$

Output: a set of elements $V := \{u'_1, \ldots, u'_k\}$ with $\text{depth}(u'_i) < \text{depth}(u'_j)$ for $i < j$ such that $\langle u_i \mid \text{depth}(u_i) \geq d\rangle \leq \langle u'_i \mid \text{depth}(u'_i) \geq d\rangle$ holds for all d and $\langle U\rangle = \langle V\rangle$.

1 **for** d *from* 1 *to* n **do**
2 **while** *there are at least two elements of depth* d *among the* u_i **do**
3 Choose an element u_j of depth d with lead of minimal absolute value $e > 0$.
4 Replace all u_i, with $i \neq j$ with $u_i \cdot u_j^k$ where $k \in \mathbb{Z}$ is a suitable integer such that d-th entry of the exponent vector of the product is less than $\text{lead}(u_j)$.

The correctness of this algorithm follows immediately if viewed in the notion of remark 2.6. The loop just implements a step-by-step Euclidean algorithm on the columns of the generator matrix. Each multiplication of step 4 is reversible thus ensuring the inclusions.

Better algorithms for computing an Hermite normal form typically do not perform the Euclidean algorithm step-by-step on the rows of the matrix. Instead one uses the Euclidean algorithm on the pivots of two rows to compute the coefficients of Bézout's identity which is also the underlying concept of the algorithm presented in [Eic21]: Given a set of generators $u_1, \ldots, u_m$ of a subgroup U of a polycyclic group one chooses two elements of the same depth d, without loss of generality say u_1 and u_2. Denote by $e_1,\ e_2$ the respective lead exponents. One then computes coefficients $a_1,\ a_2 \in \mathbb{Z}$ with $e := \gcd(e_1, e_2) = a_1e_1 + a_2e_2$. The lift $g := u_1^{a_1}u_2^{a_2}$ then has lead e and there are integers $b_1,\ b_2 \in \mathbb{Z}$ such that $g^{b_1}u_1$ and $g^{b_2}u_2$ have depth greater d. We can therefore add g to the set of generators and get

$$\langle u_1, \ldots, u_m\rangle = \langle g, u_1, \ldots, u_m\rangle = \langle g, g^{b_1}u_1, g^{b_2}u_2, u_3, \ldots, u_m\rangle.$$

The last set of generators has fewer elements of depth d while only increasing

the number of generators of higher depth making this into a finite procedure.
It is important to note that unlike in the case of integer matrices it is not sufficient to replace $u_1,\ u_2$ by g and $u_1^{c_1} \cdot u_2^{c_2}$ where $c_1,\ c_2 \neq 0$ are the coefficients of the extended Euclidean algorithm with $0 = c_1 e_1 + c_2 e_2$ as u_1 and u_2 do not necessarily commute.

2.19. Remark
As mentioned at the beginning of this chapter the problem of the algorithm in [Eic21] is its use of memory. This is caused by the necessity of adding additional elements to the list of generators. The theoretical worst case for this is that the number of elements grows exponentially in the number of generators n of the parent group. Suppose we start with m elements all of the same depth d. The algorithm now proceeds to replace two of them by one other element of the same depth d and two other elements of depth larger or equal to $d+1$. Continuing like this we will end up with $2m-1$ generators and during this calculations we computed $2(m-1) + (m-1) = 3(m-1)$ new elements. If we assume the worst case in which all elements are pairwise different and the depth only increases by one while starting from $d = 1$ we might have $m \cdot 2^{n-1} - (n-1)$ generators in the last step, when all of them will just collapse to a single one.

A first idea to prevent this from happening is to work with multiple elements at once. Given $u_1, \ldots, u_m$ of the same depth d one can compute $a_1, \ldots, a_m \in \mathbb{Z}$ such that

$$e := \gcd(\operatorname{lead}(u_1), \ldots, \operatorname{lead}(u_m)) = a_1 \operatorname{lead}(u_1) + \ldots + a_m \operatorname{lead}(u_m).$$

One can then lift the result back to the polycyclic group and compute a product

$$g := \prod_{i=1}^{m} u_i^{a_i} := u_1^{a_1} \cdot \ldots \cdot u_m^{a_m}$$

and $\operatorname{lead}(g)$ will be equal to e and thus divide $\operatorname{lead}(u_i)$ for all $i = 1, \ldots, m$. Hence there are $b_1, \ldots, b_m \in \mathbb{Z}$ such that $\operatorname{depth}(g^{b_i} \cdot u_i) > \operatorname{depth}(u_i)$. Note: As polycyclic groups are not necessarily abelian the product $\prod u_i^{a_i}$ depends on the order of the u_i. However, due to 1.9 for any ordering of the factors the product will have the same lead exponent (modulo exponent relation) and are therefore equally valid. For the purpose of readability we just fixed the ascending ordering of the index as above.
The resulting algorithm 2.20 is a generalization of Bradley's algorithm [Bra71] for Hermite normal form computations on integer matrices and avoids computing too many additional generators. Instead for each possible depth there is at most one new element computed, bounding the number of added elements

by n. However, it is empirically known that Bradley's algorithm suffers like many other Hermite normal form algorithms from coefficient explosion in the intermediate steps [Fru77]. Other algorithms like [KB79] might be polynomial bounded but when translating them into the polycyclic setting they suffer the same problem as [Eic00] and [Eic21] as they generate too many new elements.

Algorithm 2.20: Echelon form - Bradley's algorithm

Input: a non-empty set of elements $U := \{u_1, \ldots, u_m\}$ of a polycyclic group G given by a polycyclic presentation on n generators

Output: a set of elements $V := \{u'_1, \ldots, u'_k\}$ in echelon form such that $\langle u_i \mid \text{depth}(u_i) \geq d\rangle \leq \langle u'_i \mid \text{depth}(u'_i) \geq d\rangle$ holds for all d and $\langle U\rangle = \langle V\rangle$.

1 Initialize V as the empty set.
2 **for** d *from* 1 *to* n **do**
3 Set $S_d := \{u \in U \mid \text{depth}(u) = d\}$.
4 If S_d is empty, continue with the next d.
5 Assume the elements in S_d are $x_1, \ldots, x_{|S_d|}$.
6 Compute integers $a_1, \ldots, a_{|S_d|}$ with

$$\gcd(\text{lead}(x_1), \ldots, \text{lead}(x_{|S_d|})) = a_1 \,\text{lead}(x_1) + \ldots + a_{|S_d|} \,\text{lead}(x_{|S_d|})$$

7 Compute $v_d := \prod_{i=1}^{|S_t|} x_i^{a_i}$ and add it to V.
8 Replace all $u \in U$ with $\text{depth}(u) = d$ by $v_j^{-e_j} \cdot u$ where $e_j = \frac{\text{lead}(u)}{\text{lead}(v_j)}$.
9 Return V.

Our approach therefore is to use a compromise between those approaches: The idea is to partition the polycyclic presentation of the group G in such a way that one obtains a normal subgroup N such that G/N is abelian. In this quotient we will then use the well understood methods for integer Hermite normal form computation to compute coefficients which will then be used to lift the result back into the original group. This is the goal of the following discussion.

2.21. Definition
Let n be a natural number. We say P is a *segmentation* of $M = \{1, \ldots, n\}$ if it is a partition of M, i.e. $P = \{P_1, \ldots, P_k\}$ with $\emptyset \neq P_i \subset M$, $\bigcup_i P_i = M$ and $P_i \cap P_j = \emptyset$ for $i \neq j$, such that for all $i \neq j$ we have

$$\max(P_i) < \min(P_j) \text{ or } \max(P_j) < \min(P_i).$$

This relation defines a strict total order on P so we just assume $P_i < P_j$ for $i < j$.

If G is a polycyclic group given by a sequence $G = G_1 \triangleright \ldots \triangleright G_{n+1} = 1$ we say P is an *abelian segmentation* of G, if it is a segmentation of $\{1, \ldots, n\}$ and for

all $P_i \in P$ we have

$$G'_{\min(P_i)} \leq G_{\max(P_i)+1},$$

or in other words, if $G_{\max(P_i)+1}$ is a normal subgroup of $G_{\min(P_i)}$ and the quotient group is abelian.

Note that a segmentation $P = \{P_1, \ldots, P_k\}$ is uniquely determined by the maximum of the P_i for $i = 1, \ldots, k-1$. Hence there are $\binom{n-1}{k-1}$ segmentations of $\{1, \ldots, n\}$ into k subsets. The number of abelian segmentations depends on the group's relations. However, every polycyclic group G with polycyclic sequence $G = G_1 \triangleright \ldots \triangleright G_{n+1} = 1$ has at least one abelian segmentation namely $\{\{1\}, \{2\}, \ldots, \{n\}\}$. In section 2.6 we will look at the problem of finding other, preferably smaller abelian segmentations and compare them by their impact on the computations.

Given a polycyclic group together with an abelian segmentation one can immediately find such a segmentation for subgroups and quotients.

2.22. Lemma
Let G be a polycyclic group with a polycyclic series $G = G_1 \triangleright \ldots \triangleright G_{n+1} = 1$. Denote by P an abelian segmentation of G. Further, let U be a subgroup of G with an induced generating sequence $U = U_1 \triangleright \ldots \triangleright U_{k+1} = 1$. Write

$$f : \{1, \ldots, k\} \to \{1, \ldots, n\},\ i \mapsto \max\{j \mid G_j \cap U = U_i\}$$

for the map that associates the subgroups in the sequence of U to the ones in the sequence of G. Then we have:

1. One obtains an abelian segmentation P' of U by setting
$$P' := \{f^{-1}P_i \mid P_i \in P, f^{-1}P_i \neq \emptyset\}.$$
2. The minimal size of an abelian segmentation for the subgroup U is at most as large as the minimal size of an abelian segmentation of G.

Proof. The map f is by the definition of an induced sequence injective and strictly increasing. This implies that P' is indeed a segmentation of $\{1, \ldots, k\}$. Let $P'_i \in P'$ and set $a := \min P'_i,\ b = \max P'_i$. By definition there is a $P_j \in P$ such that $fP'_i \subseteq P_j$. Thus we obtain

$$U_a = U \cap G_{f(a)} \leq G_{\min P_j} \cap U$$

and

$$U_b = U \cap G_{f(b)} \geq G_{\max P_j+1} \cap U.$$

Putting this together we obtain

$$U'_a \leq G'_{\min P_j} \cap U \leq G_{\max P_j+1} \cap U \leq U_b$$

and P' is an abelian segmentation of U.

The second part immediately follows from the first one. □

2.23. Lemma
Let G be a polycyclic group with a polycyclic series $G = G_1 \triangleright \ldots \triangleright G_{n+1} = 1$ and N denote a normal subgroup of G. Denote by

$$Q_1 \triangleright \ldots \triangleright Q_{k+1} = 1$$

the polycyclic series for G/N that consists of all G_iN/N where we have removed duplicates. If we write

$$f : \{1, \ldots, n\} \to \{1, \ldots, k\},\ i \mapsto j \text{ with } Q_j = G_iN/N,$$

we have:

1. For an abelian segmentation P of G we set

   $$P'_1 := f(P_1),\ P'_{i+1} = f(P_{i+1}) \setminus (P'_1 \cup \ldots \cup P'_i).$$

   Then $P' = \{P'_i \mid 1 \leq i \leq n,\ P'_i \neq \emptyset\}$ is an abelian segmentation of G/N.

2. The minimal size of an abelian segmentation for the quotient G/N is at most as large as the minimal size of an abelian segmentation of G.

Proof. Analogue to 2.22. □

2.24. Remark
A group G is polycyclic if and only if there is a normal subgroup N that is polycyclic such that G/N is also polycyclic (see remark 1.11). One can therefore flip lemma 2.22 and 2.23 around to get a polycyclic sequence of the group G with an abelian segmentation derived from one of N and G/N. If one sets N to the derived subgroup, one can inductively conclude that the minimal possible size of a segmentation for the polycyclic group G is the length of its derived series. We look at this in more detail in example 2.41.

We will now describe the algorithm for computing an echelon form $u'_1, \ldots, u'_k$ from a given set of group elements $u_1, \ldots, u_m$ which also preserves the slightly stronger inclusion condition

$$\text{for all } u_i \text{ exist } a_1, \ldots, a_k \in \mathbb{Z} \text{ such that } u_i = u_1'^{a_1} \cdot \ldots \cdot u_k'^{a_k} \tag{1}$$

This implies the usual $\langle u_i \mid \text{depth}(u_i) \geq d\rangle \leq \langle u'_i \mid \text{depth}(u'_i) \geq d\rangle$ condition. The purpose of strengthening this condition is to give a different termination criterion for the complete induced sequence algorithm later in section 2.4.
In order to keep the proof of correctness simpler we will do this in two steps. First we assume that the given polycyclic group G is poly-C_∞ as exponent relations introduce some technical complications. After a short example we

will then state the general algorithm and make the necessary adaptations to the proof.

Algorithm 2.25: Echelon form - simplified version

Input: a non-empty set of elements $U := \{u_1, \ldots, u_m\}$ of a polycyclic group G given by a polycyclic presentation on n generators $g_1, \ldots, g_n$ with $\mathrm{rord}_i(G) = \infty$ for all i

Output: a set of elements $V := \{u'_1, \ldots, u'_k\}$ with $\mathrm{depth}(u'_i) < \mathrm{depth}(u'_j)$ for $i < j$ such that for all u_i there exists $a_1, \ldots, a_k$ with $u_i = u_1'^{a_1} \cdot \ldots \cdot u_k'^{a_k}$ and we have $\langle U \rangle = \langle V \rangle$

```
Initialize V as the empty set.
Compute a segmentation P = {P_1, ..., P_l} of G.
for t from 1 to l do
    Set S_t := {u ∈ U | depth(u) ∈ P_t}.
    Assume the elements in S_t are x_1, ..., x_{|S_t|} and write the exponent
     vectors of the elements as rows in a matrix M.
    Discard the columns of M with index not in P_t.
    Compute a (row) Hermite normal form H of M with transformation
     matrix T.
    for j from 1 to rk(H) do
        Compute v_j := ∏_{i=1}^{|S_t|} x_i^{T_{j,i}} and add it to V.
        Replace all u ∈ U with depth(u) = depth(v_j) by v_j^{-e_j} · u where
         e_j = lead(u) / lead(v_j).
Return V.
```

Proof. Termination is clear. For the correctness we show the following:

1. *Over the course of the algorithm $\langle U \cup V \rangle$ remains unchanged.* Whenever a new element is introduced to V it is a product of the $x_i \in S_t \subseteq U$ and it doesn't enlarge the generated group. On the other hand replacing an element $u \in U$ by any product $v^k u$ for $v \in V$ and $k \in \mathbb{Z}$ does not alter the subgroup as well. At the beginning of the algorithm we have $\langle U \cup V \rangle = \langle U \rangle$. So we have to show that at the end of the algorithm U only contains the neutral element of the group. For this we show:

2. *At the end of each iteration of the outer loop we have*

$$\mathrm{depth}(u) > \max P_t \text{ for all } u \in U.$$

Clearly, at the start of the first iteration we have $\mathrm{depth}(u) \geq \min P_1 = 1$ and inductively we can assume that

$$\mathrm{depth}(u) \geq \min P_t \text{ for all } u \in U$$

at the beginning of each iteration as $\max P_t = \min P_{t+1} - 1$. We then take all elements of $x_i \in U$ with depth at most $b := \max P_t$ and set $a := \min P_t$. As G_a/G_{b+1} is free abelian the matrix M contains the exponent vectors of the $x_i G_{b+1}$ in G_a/G_{b+1}. Computing an Hermite normal form of M with a transformation matrix T yields a set of generators of $\langle U \rangle G_{b+1}/G_{b+1}$ in upper triangular form. Lifting those back to G_a with the coefficients in the transformation matrix T yields the v_j. As the v_j span the quotient there exists $a_1, \ldots, a_{\mathrm{rk}(H)} \in \mathbb{Z}$ such that

$$u \equiv v_1^{a_1} \cdot \ldots \cdot v_{\mathrm{rk}(H)}^{a_{\mathrm{rk}(H)}} \mod G_{b+1}.$$

Step 10 then successively divides off these powers and after the inner loop we have $u \in G_{b+1}$ for all $u \in U$. As $\max P_l = n$ it follows that at the end of the algorithm U is trivial.

3. *The elements of V are strictly ordered by depth.* This holds true by construction: All v_j constructed during an iteration lie in the respective G_a and their class in the quotient group is non-trivial. Therefore $a \leq \mathrm{depth}(v_j) \leq b$ or simply $\mathrm{depth}(v_j) \in P_t$. As all g_i have infinite relative order the exponent vector of v_j at the positions from a to b is equal to the exponent vector of $v_j G_{b+1}$ in G_a/G_b which is just the j-th row of H. By definition H is in upper triangular form thus the v_j all have pairwise different depth in P_t and since the sets in the segmentation are pairwise disjoint the claim follows.

4. *All elements of the input can be written as $u_1'^{a_1} \cdot \ldots \cdot u_k'^{a_k}$ for some $a_i \in \mathbb{Z}$ and $V = \{u_1', \ldots, u_k'\}$.* This is trivial as we just have shown that at the end U just contains the neutral element. As any element u in U is consecutively replaced by some $u_i^{-a_i} u$ we have $1 = u_k^{-a_k} \cdot \ldots \ldots u_1^{-a_1} u$ and the claim follows. □

2.26. Example
Let G be given by

$$\langle g_1, g_2, g_3, h_1, h_2, h_3 \mid h_j^{g_i^{\pm 1}} = h_j^{-1},\ g_j^{g_i^{\pm 1}} = g_j,\ h_j^{h_i^{\pm 1}} = h_j \text{ for } 1 \leq i, j \leq 3 \rangle$$

and say we want to compute an echelon form of the subgroup generated by $u_1, \ldots, u_5$ given by

$$U = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 1 \\ -2 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 2 & 1 & 0 & 0 \\ 0 & 2 & 4 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \end{pmatrix}.$$

A segmentation of G would be $\{\{1,2,3\},\{4,5,6\}\}$. So looking at the first block we get the matrix

$$M = \begin{pmatrix} 1 & 1 & 0 \\ -2 & 0 & 0 \\ 0 & 0 & 2 \\ 0 & 2 & 4 \end{pmatrix}$$

which has Hermite normal form H with transformation matrix T:

$$H = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \\ 0 & 0 & 0 \end{pmatrix} \quad T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 2 & 1 & 2 & -1 \end{pmatrix}.$$

Computing the lifts and reducing the other generators yields

$$\left(\frac{V}{U}\right) = \left(\begin{array}{cccccc} 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 2 & 0 & 0 & 1 & 1 \\ 0 & 0 & 2 & 1 & 0 & 0 \\ \hline 1 & 1 & 0 & 0 & 0 & 1 \\ -2 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 2 & 1 & 0 & 0 \\ 0 & 2 & 6 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \end{array}\right) \rightsquigarrow \left(\frac{V}{U}\right) = \left(\begin{array}{cccccc} 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 2 & 0 & 0 & 1 & 1 \\ 0 & 0 & 2 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & -3 & -1 & -1 \\ 0 & 0 & 0 & 2 & 0 & 0 \end{array}\right)$$

(We have written the lifts on top of the original matrix to indicate the similarity to the regular Hermite normal form computation and that the group generated by U and V does not change.) The next block works just the same and we end up with the result

$$\left(\frac{V}{U}\right) = \left(\begin{array}{cccccc} 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 2 & 0 & 0 & 1 & 1 \\ 0 & 0 & 2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 2 & 2 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right).$$

Now we state the complete algorithm:

Algorithm 2.27: Echelon form

Input: a non-empty set of elements $U := \{u_1, \ldots, u_m\}$ of a polycyclic group G given by a polycyclic presentation on n generators $g_1, \ldots, g_n$

Output: a set of elements $V := \{u'_1, \ldots, u'_k\}$ with $\text{depth}(u'_i) < \text{depth}(u'_j)$ for $i < j$ such that for all u_i there exists $a_1, \ldots, a_k$ with $u_i = u'^{a_1}_1 \cdot \ldots \cdot u'^{a_k}_k$ and we have $\langle U \rangle = \langle V \rangle$

1 Initialize V as the empty set.

2 For each exponent relation of the form $g_j^{e_j} = g_{j+1}^{e_j+1} \ldots g_n^{e_n}$ set

$$r_j := (0, \ldots, 0, -e_j, e_{j+1}, \ldots, e_n) \in \mathbb{Z}^n.$$

3 Compute a segmentation $P = \{P_1, \ldots, P_l\}$ of G.

4 **for** t *from* 1 *to* l **do**

5 Set $S_t := \{u \in U \mid \text{depth}(u) \in P_t\}$.

6 Choose some ordering on $S_t = \{x_1, \ldots, x_r\}$ and write the exponent vectors of the elements as rows in a matrix M.

7 Append r_j as an additional row to M for $j \in P_t$. Discard the columns of M with index not in P_t.

8 Compute a (row) Hermite normal form H of M with transformation matrix T.

9 **for** j *from* 1 *to* $\text{rk}(H)$ **do**

10 Compute $v_j := \prod_{i=1}^{|S_t|} x_i^{T_{j,i}}$ and add it to V.

11 **if** *the column of the pivot in row j does not correspond to a generator with finite relative order* ***or*** *the pivot is less than that relative order* **then**

12 Replace all $u \in U$ with $\text{depth}(u) = \text{depth}(v_j)$ by $u \cdot v_j^{-e_j}$ where $e_j = \frac{\text{lead}(u)}{\text{lead}(v_j)}$.

13 Add for all $j > |S_t|$ the element $\prod_{i=1}^{|S_t|} x_i^{T_{j,i}}$ to U if it is not the neutral element.

14 **else**

15 Add v_j to U if it is not the neutral element.

16 Return V.

Proof. We discuss the necessary changes to the steps 1 to 3 of the proof of 2.25. Step 4 carries over.

1. As before $\langle U \cup V \rangle$ stays unchanged over the entire algorithm. The only difference is that the v_j are either added to U or to V.

2. Using the notation of before we write $a := \min P_t$ and $b := \max P_t$. We

only need to look at the case when G_a/G_{b+1} is not free abelian. In this case M contains not only the exponent vectors of the x_iG_{b+1} but also a non-zero row corresponding to $1G_{b+1}$ for each exponent relation of the form

$$r_i := (0, \ldots, 0, -e_i, e_{i+1}, \ldots, e_b) \tag{2}$$

with $a \leq i \leq b$. Computing an Hermite normal form therefore gives a preimage of a generating set of $\langle S_t\rangle G_{b+1}/G_{b+1}$ in $\mathbb{Z}^{b+1-a}$ where the pivots of the columns corresponding to a finite relative order are less or equal to that order as (2) lies in the row span. Call the j-th row of the Hermite normal form H_j. Assume the pivot of row j is equal to the the relative order. As (2) lies in the row span there are $a_{j+1}, \ldots, a_{\mathrm{rk}\,H}$ such that

$$r_i = -H_j + a_{j+1}H_{j+1} + \ldots + a_{\mathrm{rk}\,H}H_{\mathrm{rk}\,H}.$$

Mapping this back to the quotient it follows that

$$v_j \equiv v_{j+1}^{a_{j+1}} \cdot \ldots \cdot v_{\mathrm{rk}\,H}^{a_{\mathrm{rk}\,H}} \mod G_{b+1}. \tag{3}$$

It follows that the quotient $\langle S_t\rangle G_{b+1}/G_{b+1}$ is already generated by the elements corresponding to the rows of H where the pivot is smaller than the relative order. The elements in U can now be reduced to elements of depth greater b as before. For the elements added in step 15 this holds true as well due to equation (3).

3. The elements that are added to V correspond to rows in H where the pivot is smaller than the respective relative order. Hence the depth of said element in G_a/G_{b+1} is equal to the pivot index. The rest of the argument in the proof of 2.25 carries over. □

2.28. Remark
The most costly step of algorithm 2.27 is when the coefficients of the transformation matrix are used to calculate the lift $v_j = \prod x_i^{T_{j,i}}$. Even if the resulting v_j has a small exponent vector, the powers of the x_i do not need to be small, making their computation painful especially if the coefficients in T are large. However, the transformation matrix T of the matrix M is in most cases not unique as the number of generators m of the input is typically larger than the maximal number of generators n. This allows us to try to compute a transformation matrix with small coefficients in step 8. An algorithm for the computation of a Hermite normal form with small transformation matrix is presented in [HM98] where they use a lattice reduction with the well-known algorithm by Lenstra, Lenstra and Lovász (see [Coh93, cp. 2.6.1]). An implementation for this is given in GAP in the EDIM [EDIM] package which we utilized in our algorithm. Using lattice reduction to compute a small transformation matrix

works best if there are enough generators i.e. rows in M which are linearly dependent. This has the interesting side effect that adding more redundant generators as discussed in example 2.15 can often improve the performance as the lattice reduction is able to produce smaller transformation matrix. Later in example 2.40 we will give some runtime results which show this in practice.

2.29. Remark
Let u be an element of the input of algorithm 2.27 and write $d := \text{depth}(u)$. Set $u'_1, \ldots, u'_k$ as the output. As there are $a_1, \ldots, a_k \in \mathbb{Z}$ such that $u = u_1'^{a_1} \cdot \ldots \cdot u_k'^{a_k}$ we can apply remark 1.8 and deduce that $a_i = 0$ for all i with $d > \text{depth}(u'_i)$. Furthermore there has to be a u'_i with $\text{depth}(u'_i) = d$. Applying lemma 1.9 we know that $\text{lead}(u'_i)$ needs to divide $\text{lead}(u)$ modulo the relative order. Since we work with the lift over the integers it follows that $\text{lead}(u'_i) \leq \text{lead}(u)$. We will make use of this fact later.

There are a few technical improvements that can be done in order to further improve the algorithm's performance. We list those with the greatest impact on the computations:

- The last segment defines an abelian subgroup. Therefore one can omit the computation of the transformation matrix together with the lattice reduction as no lifting is necessary. This actually has a great impact on the performance as the final segment is where the growth of the coefficients accumulates making the final computation more costly than the others.
- Using algorithm 2.12 in between to reduce the size of the coefficients is usually more costly. However in the case that there is a l such that the last $n - l$ generators $g_{l+1}, \ldots, g_n$ of the parent group are contained in U one can directly skip any computations for elements of depth greater l as well as reducing all generators of smaller depth. This also works together with the previous adaptation.
- Keeping the list of generators sorted by depth during the whole algorithm allows for faster access to the elements of certain depths in the loop.

2.4. Termination Criterion

In this section we want to state an alternative criterion as halting condition for the induced sequence. As mentioned before, the termination of the algorithm is based on the fact that in polycyclic groups every ascending sequence of

subgroups becomes stationary eventually. We apply this fact to the sequences of subgroups

$$\langle u_i \mid \text{depth}(u_i) \geq d \rangle \leq \langle u_i' \mid \text{depth}(u_i') \geq d \rangle \tag{4}$$

for each possible depth d before and after each round of echelon form computation. However the difficulty in this lies in recognizing whether the sequence has become stationary as comparing subgroups itself requires the computation of (canonical) induced sequences. There are multiple options to circumvent this problem, the simplest one being to always compute canonical sequences. As they are unique one can check the equality in (4) simply by checking for changes in the generator set. For the more general version of the algorithm [Eic21] argued that the added commutators only increase in depth. However, this argument only works for some polycyclic groups (e.g. nilpotent groups, see chapter 4). A counterexample can be constructed using the group of example 2.15. In [HEO05] every candidate for a new generator is first tested whether it can be expressed in terms of the old ones which will always work if the sequence is already an induced one by definition 2.1 and 1.5. This approach requires a lot of computational effort and isn't applicable in our case as we are handling multiple elements at once instead of each of them individually. This motivates the following discussion where we are going to use the stronger inclusion condition of equation (1) on page 31 to give an alternative criterion by looking at the lead exponents.

First we show:

2.30. Lemma
Let G be a polycyclic group and $u_1, \ldots, u_k, u_1', \ldots u_l'$ be elements of G such that $\text{depth}(u_i) < \text{depth}(u_j)$ for all $i < j$ and $\text{depth}(u_i') < \text{depth}(u_j')$ for all $i < j$. Assume we have

$$\langle u_i \mid \text{depth}(u_i) \geq d \rangle \leq \langle u_j' \mid \text{depth}(u_j') \geq d \rangle$$

for all d. Then equality holds for all d if

1. $l = k$ and
2. ${u_i'}^{-1} \cdot u_i$ is in $\langle u_j' \mid \text{depth}(u_j') > \text{depth}(u_i) \rangle$ for all i.

Proof. Assume that the conditions are fulfilled but there is a d such that

$$\langle u_i \mid \text{depth}(u_i) \geq d \rangle \lneq \langle u_i' \mid \text{depth}(u_i') \geq d \rangle.$$

Without loss of generality choose d to be maximal and set i to the index with $\text{depth}(u_i) = d$. Now as $l = k$ this implies that for all i we have $\text{depth}(u_i) = \text{depth}(u_i')$. Then from our assumption and because d was chosen maximal it follows

$${u_i'}^{-1} \cdot u_i \in \langle u_j' \mid \text{depth}(u_j') > d \rangle = \langle u_j \mid \text{depth}(u_j) > d \rangle.$$

So $u_i' \in \langle u_j \mid \text{depth}(u_j) \geq d \rangle$ which contradicts the assumption that

$$\langle u_i \mid \text{depth}(u_i) \geq d \rangle \neq \langle u_i' \mid \text{depth}(u_i') \geq d \rangle. \qquad \square$$

The converse is of course not true. Taking a finite cyclic group given by two different generators which are not inverse to each other already yields a counterexample.

2.31. Definition
Let G be a polycyclic group given by a polycyclic sequence and let $u, u_1, \ldots, u_k$ be elements of G with $\text{depth}(u_i) < \text{depth}(u_j)$ for all $i < j$. Then we say that u has a *normal form with respect to* $u_1, \ldots, u_k$ if there are integers $a_1, \ldots, a_k$ with

$$u = u_1^{a_1} \cdot \ldots \cdot u_k^{a_k}$$

where all a_i are zero for $\text{depth}(u_i) < \text{depth}(u)$.

If the $u_1, \ldots, u_k$ form an induced sequence then every element in $\langle u_1, \ldots, u_k \rangle$ can be written in normal form with respect to the u_i. (See definition 1.5.) This notation now allows us to state our halting condition:

2.32. Proposition
Let G be a polycyclic group given by a polycyclic sequence and let $u_1, \ldots, u_k$ and $u_1', \ldots u_l'$ be elements of G such that $\text{depth}(u_i) < \text{depth}(u_j)$ for all $i < j$ and $\text{depth}(u_i') < \text{depth}(u_j')$ for all $i < j$. Assume we have that all ${u_i'}^{-1} \cdot u_i$ have a normal form with respect to $u_1', \ldots u_l'$. Then for all d we have

$$\langle u_i \mid \text{depth}(u_i) \geq d \rangle = \langle u_j' \mid \text{depth}(u_j') \geq d \rangle$$

if we have

1. $l = k$ and
2. $\text{lead}(u_i) = \text{lead}(u_i')$ for all $1 \leq i \leq k$.

If in addition $u_1, \ldots, u_k, u_1', \ldots u_l'$ are all normalized, the converse also holds.

Proof. The first part follows from lemma 2.30. If $\text{lead}(u_i) = \text{lead}(u_i')$ then we have ${u_i'}^{-1} \cdot u_i \in G_{d+1}$ where d denotes the depth of u_i. This product has a normal form with respect to the u_j' by assumption and therefore lies in $\langle u_j' \mid \text{depth}(u_j') > d \rangle$.
For the converse assume all elements are normalized. Clearly, we have $l = k$ or there would be a d with

$$\langle u_i \mid \text{depth}(u_i) \geq d \rangle \subsetneq \langle u_j' \mid \text{depth}(u_j') \geq d \rangle.$$

For the second condition note that for all i we have

$$u_i \in \langle u'_i, \ldots, u'_k \rangle \text{ and } u'_i \in \langle u_i, \ldots, u_k \rangle.$$

Applying lemma 1.9 and remark 2.8 about normalized elements and we get

$$\text{lead}(u_i) \in \langle \text{lead}(u'_i) \rangle \text{ and } \text{lead}(u'_i) \in \langle \text{lead}(u_i) \rangle.$$

As both leads are positive we get $\text{lead}(u_i) = \text{lead}(u'_i)$. □

2.5. The Complete Algorithm

We can now summarize our results and state the new algorithm for computing induced sequences of polycyclic groups.

Algorithm 2.33: Induced sequence

Input: a finite set of elements M of a polycyclic group G given by a polycyclic sequence $G = G_1 \triangleright \ldots G_n \triangleright G_{n+1} = 1$

Output: an induced generating sequence for the subgroup $U = \langle M \rangle$

```
1  Initialize X as an empty list.
2  Use algorithm 2.27 with lattice reduction to compute a set of generators
     in echelon form from M and store them in a list L.
3  Initialize an empty list L_2.
4  for d from 1 to n do
5      Set g to the element of smallest depth in L and remove it from L.
6      Compute a normalization g' of g and insert it into X. Insert
         g'^(-lead(g)/lead(g')) · g into L_2.
7      if r_d := rord_d(G) < ∞ then
8          Add g'^(r_d/lead(g')) to L_2.
9      if L and L_2 are empty then
10         Return X.
11     Set condition to true.
12     while condition do
13         For all h ∈ L add g'^(-1) h g' to L_2.
14         Use algorithm 2.27 with lattice reduction to compute a set of
             generators in echelon form from L ∪ L_2 and replace L_2.
15         if length of L equals the length of L_2 and for all x ∈ L, y ∈ L_2
             with depth(x) = depth(y) we have lead(x) = lead(y) then
16             Set condition to false.
17         Replace L by L_2 and L_2 by an empty list.
```

Proof. The first echelon computation is just to reduce the initial number of generators and to simplify the proof. We can now assert that $X \cup L \cup L_2$ contains a set of generators of U during the whole algorithm: In the beginning of the algorithm this is true as X and L_2 are empty and L contains the generator set. Whenever an element from L is removed its normalization is inserted into X as well as the product $g'^{-\frac{\text{lead}(g)}{\text{lead}(g')}} \cdot g$. Thus, the group generated by $X \cup L \cup L_2$ does not change. Afterwards, whenever an element is inserted into L_2 it is already contained in $\langle X \cup L \rangle$ and therefore does not change the group. The replacement steps in line 14 and 17 do not change the generated group either as algorithm 2.27 also outputs a generator set. Hence, when the algorithm returns X it must be a set of generators for U.

Furthermore, it has to be in echelon form as one shows analogously that $X \cup L$ is always in echelon form because we have $\max\{\text{depth}(u) \mid u \in X\} < \min\{\text{depth}(u) \mid u \in L\}$ at all times. Now we need to show that X contains an induced sequence at the end of the algorithm. For this we use 2.3 and show that after the d-th iteration of the outer for-loop where we have added g' to X for all $h \in U_{d+1}\langle u \in X \cup L \mid \text{depth}(u) > d \rangle$ also $g'^{-1}hg' \in U_{d+1}$. The inner while-loop ends when the conditions of Lemma 2.32 are met as we have shown that algorithm 2.27 outputs generators such that the elements in $L \cup L_2$ can be written in normal form with respect to the output. So by applying this we get that the loop ends when adding conjugates no longer enlarges U_{d+1}. We do not need to show 2 as we explicitly add this element in line 8. As U_{d+1} then does not change anymore (with the same argument as above for $\langle X \cup L \cup L_2 \rangle$) we know that this stays true until the end of the algorithm and the output is therefore an induced sequence.

The only thing left is to show that the algorithm actually terminates, i.e. the halting condition is met at some point in each iteration of the for-loop. But this is the case as we have seen in remark 2.29 that applying algorithm 2.27 the (positive) lead exponents of the generators can only decrease or new generators with a depth not yet present in the input can be added. As both can only happen finitely many times, at some point both the number of generators as well as the lead exponents no longer change and proposition 2.32 tells us that the sequence has become stationary. □

2.6. Minimal Segmentations

In this section we want to look at how to determine a "good" abelian segmentation. Every polycyclic group G with polycyclic series $G = G_1 \triangleright \ldots \triangleright G_{n+1} = 1$ has the trivial segmentation $\{\{1\}, \{2\}, \ldots, \{n\}\}$. However, using this segmen-

tation causes algorithm 2.27 to operate like the Bradley algorithm 2.20. So we would like to reduce the number of subsets in the segmentation. We present two algorithms for this and compare them with respect to their impact on the runtime of 2.33.

2.34. Lemma
Let G be a polycyclic group given by a polycyclic presentation on n generators $g_1, \ldots, g_n$. Then the derived subgroup is generated by

$$\{[g_i, g_j] := g_i^{-1} g_j^{-1} g_i g_j \mid 1 \leq i, j \leq n\}.$$

Proof. This is a special case of [Eic00, 4.17]. □

Algorithm 2.35: Abelian segmentation (top down)

Input: a polycyclic group G given by a polycyclic presentation on generators $g_1, \ldots, g_n$
Output: an abelian segmentation of G

```
1 Initialize i as 1 and P = ∅.
2 while i < n + 1 do
3     Set j = i and b = n + 1.
4     while j < b do
5         Set b = min({b} ∪ {depth([g_j^±1, g_k^±1]) | j < k ≤ n}).
6         Increment j by 1.
7     Add {i, ..., j − 1} to P and set i to j.
8 Return P.
```

2.36. Lemma

1. Algorithm 2.35 terminates with an abelian segmentation.
2. The returned segmentation P is minimal, i.e. for all abelian segmentations Q of G holds $|P| \leq |Q|$.

Proof.

1. Termination is easy: As in every iteration of the outer loop b is reset to $n + 1$ the inner while loop will always be entered. Therefore j is incremented at least once which causes i to strictly increase in every loop iteration.

   For the correctness we need to show that every time we add $\{i, \ldots, j-1\}$ to P we have that G_i/G_j is abelian or equivalently $G_i' \leq G_j$. So assume we are at the end of such an iteration and fix the corresponding i and j. By lemma 2.34 the commutator subgroup G_i' is generated by the $[g_l^{\pm 1}, g_k^{\pm 1}]$

with $i \leq l < k$. Of course $[g_l^{\pm 1}, g_k^{\pm 1}] \in G_j$ for all $j \leq l < k$. Now assume that there is a commutator x, say without loss of generality $x = [g_l, g_k]$, with $l < j$ such that $\text{depth}(x) < j$. Then at one point of the inner loop this generator would have been observed and therefore $b \leq \text{depth}\, x$. However, the loop ensures that at all time $j \leq b$ holds. So this results into a contradiction. Therefore all commutators generating G_i' have depth at least j and so $G_i' \leq G_j$.

2. By construction the algorithm fixes a set $\{i, \ldots, j-1\}$ in the segmentation when it has found a commutator of depth j. It therefore chooses the maximal index j such that $G_i' \leq G_j$. So we have to show that this "greedy" approach results in a minimal segmentation. Say the algorithm outputs the segmentation $P = \{P_1, \ldots, P_k\}$ and assume $Q = \{Q_1, \ldots, Q_l\}$ is an abelian segmentation of minimal size. As usual assume that the P_i and Q_i are ordered in the obvious way. If we replace Q by $\{P_1, Q_2 \setminus P_1, \ldots, Q_l \setminus P_1\}$ we still have an abelian segmentation of minimal size: $Q_1 \subseteq P_1$ as $1 \in P_1$ and P_1 was by construction maximal in size. Additionally, each of the sets $Q_i \setminus P_1$ is non-empty or by excluding them we would obtain a segmentation of smaller size contradicting the choice of Q. As $Q_2 \setminus P_1$ is non-empty it even follows that $Q_i = Q_i \setminus P_1$ for $i > 2$ due to the ordering of the sets. If we continue inductively we can insert every set of P into a minimal segmentation. Hence P itself must be minimal. □

2.37. Remark
The optimality of the result can also be obtained using graph theory by modelling the group as an interval graph: We set $\mathcal{G} = (V, E)$ where $V = \{1, \ldots, n\}$ is the set of nodes and

$$E = \{(i, j) \mid i < j,\ G_i' \leq G_j\}$$

the set of edges. The corresponding set of intervals is

$$\mathcal{I} = \{[i, j(i) - 1] \mid 1 \leq i \leq n\} \text{ where } j(i) = \max\{j > i \mid G_i' \leq G_j\}.$$

If $(i, j) \in E$ then for any $i \leq k \leq j$ also $(k, j) \in E$ and $(i, k) \in E$. Hence, the set $\{i, \ldots, j\}$ forms a clique in the graph and any abelian segmentation defines a clique covering of the graph. While the clique covering problem for general graphs is known to be NP-hard it is like many other NP-hard problems solvable in polynomial time for chordal graphs. In the special case of interval graphs the clique covering problem becomes a special instance of an interval scheduling problem for which it is known that the greedy algorithm returns optimal solutions. For more background see [KT06] and [Kru12].

Analogously to algorithm 2.35 one can compute a segmentation of the group by working from the bottom-up.

Algorithm 2.38: Abelian segmentation (bottom up)

Input: a polycyclic group G given by a polycyclic presentation on generators $g_1, \ldots, g_n$

Output: an abelian segmentation of G

```
1 Initialize j as n, i as n − 1 and P = ∅.
2 while i > 0 do
3     Set b = min{depth([g_i^{±1}, g_k^{±1}]) | i < k ≤ n}.
4     if b < j + 1 then
5         Add {i + 1, ..., j} to P and set j to i.
6     Decrement i by 1.
7 Add {1, ..., j} to P.
8 Return P.
```

We get the same result as in 2.36:

2.39. Lemma

1. Algorithm 2.38 terminates with an abelian segmentation.
2. The returned segmentation P is minimal, i.e. for all abelian segmentations Q of G holds $|P| \leq |Q|$.

Proof. Analogue to 2.36. □

2.40. Example

The *Heisenberg group* $\mathcal{H}_n$ is a polycyclic group on $2n+1$ generators and is given by the following polycyclic presentation:

$$\langle g_1, \ldots, g_{2n+1} \mid g_{n+i}^{g_i} = g_{n+i}g_{2n+1},\ g_{n+i}^{g_i^{-1}} = g_{n+i}g_{2n+1}^{-1} \text{ for } 1 \leq i \leq n,\ g_j^{g_i^{\pm 1}} = g_j \text{ else} \rangle.$$

It is a nilpotent group (see definition 4.2) and of Hirsch number $2n+1$ which means that the exponent vector of a group element is unbounded in any component. This makes the group into an ideal candidate to observe the impact of the different possible segmentations on the performance of the induced sequence algorithm 2.27. Using 2.35 and 2.38 one computes two different minimal segmentations: The top down approach yields $\{\{1, \ldots, 2n\}, \{2n+1\}\}$ and the bottom up algorithm outputs $\{\{1, \ldots, n\}, \{n+1, \ldots, 2n+1\}\}$. For comparison we also include the trivial segmentation $\{\{1\}, \ldots, \{2n+1\}\}$. For the test we used our GAP implementation of 2.27 to compute for different n a canonical induced polycyclic sequence of a subgroup generated by m randomly chosen

generators. To get more reliable results we take the total duration of repeating this for l different test sets (where l will vary as the complexity of the example does). The following table contains the average runtime of these tests. In all three cases a standard from the left collector has been used. Unless stated otherwise all entries are given in milliseconds.

			trivial	bottom up	top down
$n = 3$	$m = 8$	$l = 10000$	2.23	1.17	8.71
$n = 3$	$m = 200$	$l = 100$	136.51	63.03	148.39
$n = 4$	$m = 6$	$l = 100$	2.91	1.81	627.02
$n = 4$	$m = 50$	$l = 100$	33.61	9.62	22.21
$n = 8$	$m = 5$	$l = 4$	6min 48s	229.80	230.00
$n = 8$	$m = 100$	$l = 10$	$*$[1]	75.00	229.80
$n = 8$	$m = 10$	$l = 1$	$*$	368.00	$*$
$n = 20$	$m = 100$	$l = 100$	$*$	575.13	574.59

Table 1: Average runtime in milliseconds rounded to two decimal places for subgroup computations on m generators in $\mathcal{H}_n$

The direct comparison of the different segmentations yield a few interesting observations: As expected the trivial segmentation reaches its best performance for smaller n and gets significantly worse for larger n. This is due to the aforementioned coefficient explosion (see discussion on algorithm 2.20) occurring in the exponent vector with increasing depth. However, it is remarkable that in some cases it outperforms the top-down approach. The latter ties with the bottom-up approach if m is comparatively small relative to n or if it is relatively large. But in some cases for m in the range between n and $2n$ the bottom-up segmentation returns results after a fraction of a second while the top-down version doesn't even terminate within hours. The reason for this is the size of the subsets in the segmentation. While the top-down algorithm produces a minimal segmentation by maximizing the size of the first subsets, the bottom-up approach achieves this by also minimizing the maximal size of each subset by spreading them more evenly. If we are in the step where algorithm 2.27 computes a small transformation matrix and if the block size is larger than the number of elements, then we look at matrices with more columns than rows. This makes it unlikely that the kernel of the row space of the matrix is sufficiently large (if it is even non-trivial) which in turn hinders the lattice reduction in finding a small transformation matrix. In cases where m is sufficiently large relative to n this isn't a problem anymore and the difference

[1] Computations aborted after 18 hours.

between bottom-up and top-down gets smaller up to even negligible as seen in table 1 in the case $n = 20$ and $m = 100$. For really small m there is almost no notable difference as for few generators there is not much to compute.

In general, the bottom-up segmentation usually performs best. However, exceptions might arise depending on the underlying group structure. E.g. in cases where exponent relations bound the size of the involved coefficients, the top-down segmentation can perform better if it produces a segmentation where the unbounded coefficients are spread more evenly.

As mentioned in remark 2.24 the number of subsets in a segmentation is bounded by the length of the derived series. However, the actual minimal size depends on the given polycyclic presentation as the following example illustrates.

2.41. Example
Let G be given by

$$\begin{aligned}\langle g_1, \ldots, g_n \mid g_j^{g_i^{\pm 1}} &= g_j && \text{for } i \equiv j \mod 2,\\ g_j^{g_i^{\pm 1}} &= g_j^{-1} && \text{for } i \not\equiv j \mod 2\rangle\end{aligned}$$

then no generator commutes with its direct neighbours. Thus, the only feasible segmentation is just the trivial one $\{\{1\}, \{2\}, \ldots, \{n\}\}$. However the commutator subgroup of G is $G' = \langle g_i^2 \mid 2 \leq i \leq n\rangle$ which is abelian so the length of the derived series is just two.

In such cases it might be advantageous to compute a different presentation for the group which refines the derived series. This can be done by computing an induced sequence of the subgroups in the derived series. In the above example one can add generators $h_2, \ldots, h_n$ with $g_i^2 = h_i$ to get the presentation

$$\begin{aligned}G = \langle g_1, \ldots, g_n, h_2, \ldots, h_n \mid g_j^{g_i^{\pm 1}} &= g_j && \text{for } i \equiv j \mod 2,\\ g_j^{g_i^{\pm 1}} &= g_j h_j^{-1} && \text{for } i < j,\ i \not\equiv j \mod 2,\\ h_j^{h_i^{\pm 1}} &= h_j && \text{for all } 1 < i, j \leq n\rangle\end{aligned}$$

Now the g_i commute modulo $\langle h_2, \ldots, h_n\rangle$. So we can give a segmentation consisting only of two sets: $\{\{1, \ldots, n\}, \{n+1, \ldots, 2n-1\}\}$

2.7. Benchmarking the new induced sequence algorithm

The author has tested the described induced sequence algorithm on various groups. We finish this chapter by giving some concrete runtime examples of these tests. All tests were run on the same machine (an Intel Core i7-8700 processor with 16 GB of memory on Kubuntu 22.04) running GAP in version 4.11 and Polycyclic 2.16. For the tests we used our own GAP implementation of algorithm 2.33 using the bottom-up segmentation of algorithm 2.38 and compared it to a modified[2] version of the code provided with [Eic21]. We did not include a comparison to the implementation in Polycyclic itself as it is known to produce wrong results. We have included an example where this is the case in the appendix.

For the tests itself we chose different groups in which we chose m random elements using Polycyclic's `Random()` function. We proceeded to compute an induced sequence generated by these elements once with algorithm 2.33 and once (with the same generators) with [Eic21]. The table 2 contains the average runtimes over l-many computations in milliseconds. The column n displays the number of generators in the series of the respective group. Computations marked with an asterisk $*$ have been aborted after the total runtime exceeded ten minutes.

Examples with smaller l should be taken with a grain of salt as the test sets are chosen randomly.

We want to note a few things: As shown in the table we have used proper subgroups in some cases to choose the random elements from. However the computations were performed in the parent group. The idea behind this was to ensure that the group generated by the randomly chosen elements will be a proper subgroup to prevent computational short-cuts as described at the end of section 2.3. This also causes the coefficients of the random generators to be larger on average which also increases the complexity.

When we have chosen a direct product we used the sequence described at the beginning of the next chapter. We have chosen this sequence as it is an easy way to effectively double the complexity of the example both in terms of generators and the length of an abelian segmentation. Note however that there are more efficient sequences for computations in the direct product.

[2]The code contained a minor indexing bug causing it to not check all commutators. We fixed this for the test.

Group	n	m	l	[Eic21]	algorithm 2.33
$\mathcal{G}_3$	3	3	1000	0.13	0.28
$\mathcal{V}_3 \leq \mathcal{G}_3$	3	100	100	5.42	11.85
$\mathcal{G}_3 \times \mathcal{G}_3$	6	6	10	5352.70	2.70
$\mathcal{G}_3 \times \mathcal{G}_3$	6	6	100	$*$	32.44
$\mathcal{V}_3 \times \mathcal{V}_3 \leq \mathcal{G}_3 \times \mathcal{G}_3$	6	6	10	$*$	1538.00
$\mathcal{G}_6$	6	6	100	83.18	0.92
$\mathcal{G}_6$	6	10	100	180.93	1.00
$\mathcal{H}_3$	7	7	100	1.86	1.03
$\mathcal{H}_4$	9	9	100	8.11	1.84
$\mathcal{H}_5$	11	11	10	1965.20	3.20
$\mathcal{H}_5$	11	11	100	$*$	3.22
$\mathcal{H}_{20}$	41	41	100	$*$	398.92
$\mathcal{H}_{20}$	41	100	100	$*$	568.78
$\mathcal{U}_{20} \leq \mathcal{H}_{20}$	41	100	100	$*$	970.02
$\mathcal{G}_{24}$	24	10	10	$*$	1663.30
$\mathcal{G}_{24}$	24	100	10	$*$	426.60
$\mathcal{G}_{24}$	24	100	100	$*$	523.01

Table 2: Average runtime in milliseconds rounded to two decimal places for l subgroup computations on m generators

On a final note: We have also experimented with Magma [MAG] but refrained from including a proper comparison here. The reason for this is that Magma uses a faster collection algorithm than the one implemented in Polycyclic. Table 2.7 contains a few runtimes it takes Polycyclic to compute the k-th power of $u = g_1^7 g_2^2 g_3^{-1}$ in $\mathcal{G}_3$. In contrast to this, Magma computed all these powers in

k	64	256	1024	2048	4096	10000
time in ms	4	15	143	552	2153	14735

less than 10ms. As collection of powers is a vital step in the algorithm we expect a big impact on the runtime. However, as we cannot inspect Magma's code we cannot quantify the precise influence.

3. Application: Computing Intersections

For polycyclic groups [Eic00] describes an algorithm to determine the intersection of two subgroups U and N. The algorithm reduces the problem to the case where N is normalized by U. In this case they use the isomorphism theorem to obtain

$$U \hookrightarrow UN \longrightarrow UN/N \cong U/U \cap N.$$

By determining the kernel of this homomorphism one can obtain the intersection. An algorithm for this is described in [Eic00] and works by computing an induced polycyclic sequence in $UN/N \times UN$.

In this section we provide an alternative way of computing the intersection $U \cap N$ when U normalizes N using an adaptation of the Zassenhaus algorithm known from linear algebra. We implement this by using the ideas of section 2 to obtain an algorithm which we will then compare to the one described above. The goal of this is to illustrate that different algorithms for the same task might profit differently from the new induced polycyclic sequence algorithm and therefore existing algorithms might need to be revisited.

For a polycyclic group G with polycyclic series

$$G = G_1 \trianglerighteq G_2 \trianglerighteq \ldots \trianglerighteq G_n \trianglerighteq 1$$

one obtains a polycyclic series of $G \times G$ by taking

$$G_1 \times G_1 \trianglerighteq G_2 \times G_1 \trianglerighteq \ldots \trianglerighteq G_n \times G_1 \trianglerighteq 1 \times G_1 \trianglerighteq 1 \times G_2 \trianglerighteq \ldots \trianglerighteq 1 \times G_n \trianglerighteq 1.$$

If $P = \{P_1, \ldots, P_k\}$ is an abelian segmentation of G, we set $P' := \{P'_1, \ldots, P'_k\}$ where $P'_i := \{n + k \mid k \in P_i\}$ is a translate of P_i. Then

$$Q := \{P_1, \ldots, P_{k-1}, P_k \cup P'_1, P'_2, \ldots, P'_k\}$$

is an abelian segmentation of $G \times G$ with respect to the above series.

We can now state:

3.1. Proposition
Let G be a group. Let U and N be subgroups of G such that N is normalized by U. Define in $G \times G$ the two subgroups

$$\tilde{U} := \{(u, u) \mid u \in U\} \quad \text{and} \quad \tilde{N} := \{(n, 1) \mid n \in N\}.$$

If $\pi_1,\ \pi_2 : G \times G \to G$ define the canonical projections, we have

$$\pi_2((1 \times G) \cap \tilde{U}\tilde{N}) = U \cap N$$

as well as

$$\pi_1(\tilde{U}\tilde{N}) = UN.$$

Proof. The statement $\pi_1(\tilde{U}\tilde{N}) = UN$ is clear. It suffices to show the statement about the intersection. First note that since U normalizes N we also have that $\tilde{U}$ normalizes $\tilde{N}$. Therefore $\tilde{U}\tilde{N}$ is a subgroup of $G \times G$. Now we have that

$$\begin{aligned}(1 \times G) \cap \tilde{U}\tilde{N} &= \{(1, y) \mid \exists u \in U,\ n \in N : (1, y) = (un, u)\} \\ &= \{(1, y) \mid y \in U \cap N\}.\end{aligned}$$

Applying π_2 we get $\pi_2((\{1\} \times G) \cap \tilde{U}\tilde{N}) = U \cap N$. □

Determining the intersection $(1 \times G) \cap \tilde{U}\tilde{N}$ can be done by computing an induced generating sequence of $\tilde{U}\tilde{N}$. By definition 2.1 $(1 \times G) \cap \tilde{U}\tilde{N}$ is given by the elements of depth larger than n. We therefore get the following algorithm:

Algorithm 3.2: Zassenhaus algorithm for polycyclic groups

Input: two subgroups $U,\ N \leq G$ of a polycyclic group given by a polycyclic sequence on n generators and U normalizes N
Output: an induced generator sequence of $U \cap N$

1 Set up the polycyclic series of $G \times G$ together with an abelian segmentation.
2 Choose a set of generators $u_1, \ldots, u_k$ of U and $n_1, \ldots, n_l$ of N.
3 Use algorithm 2.33 to compute a canonical induced generator sequence S of
$$\tilde{U}\tilde{N} = \langle \{(u_i, u_i) \mid i\} \cup \{(n_j, 1) \mid j\} \rangle.$$
4 Return the set
$$\{\pi_2(s) \mid s \in S \text{ with } \operatorname{depth}(s) \geq n\}.$$

3.3. Example
Let

$$G = \langle g_1, g_2, g_3 \mid g_3^{g_1^{\pm 1}} = g_3^{g_2^{\pm 1}} = g_3^{-1},\ g_2^{g_1^{\pm 1}} = g_2 g_3^3 \rangle$$

and $U = \langle g_1^2 g_2^3,\ g_3^3 \rangle,\ N = \langle g_1^2 g_3,\ g_2,\ g_3^2 \rangle$. Note that U normalizes N but N does not normalize U. An abelian segmentation of G is $\{\{1, 2\},\ \{3\}\}$, so one

for $G \times G$ is $\{\{1,2\},\ \{3,4,5\},\ \{6\}\}$. Using the notation of remark 2.6 (where we have indicated the segmentation by vertical separators and omitted leading zeros) we first need to compute an echelon form of $\tilde{U}\tilde{N}$:

$$\left(\begin{array}{cc|ccc|c} 2 & 3 & 0 & 2 & 3 & 0 \\ & & 3 & 0 & 0 & 3 \\ 2 & 0 & 1 & 0 & 0 & 0 \\ & 1 & 0 & 0 & 0 & 0 \\ & & 2 & 0 & 0 & 0 \end{array}\right) \longrightarrow \left(\begin{array}{cc|ccc|c} 2 & 0 & 1 & 2 & 3 & 0 \\ & 1 & 0 & 0 & 0 & 0 \\ & & 1 & 0 & 0 & 3 \\ & & & 2 & 3 & -3 \\ & & & & & 6 \end{array}\right).$$

This already defines an induced generating sequence of $\tilde{U}\tilde{N}$ and we don't need to add conjugates in this example. Reading off the elements of depth greater 3 we get $U \cap N = \langle g_1^2 g_2^3 g_3^{-3},\ g_3^6 \rangle$.

When comparing the two algorithms one can already make a few observations that might impact the performance. Both algorithms internally use an induced sequence computation over a direct product. Algorithm 3.2 works over $G \times G$ while the algorithm in [Eic00] does so over $U \times UN/N$. The former approach has the disadvantage that it doubles the number of generators as well as the Hirsch number compared to G. In contrast $U \times UN/N$ is potentially generated by less elements while also having a smaller Hirsch number. On the other hand the approach in [Eic00] requires additional work to set up: One needs to compute UN first to obtain the morphism $U \to UN/N$, while algorithm 3.2 does not need such preparations and instead computes UN as a by-product. In practice the two algorithms perform fairly similar as the following test illustrates.

3.4. Example
For the test we used the group

$$\begin{aligned} G = \langle g_1, \ldots, g_6 \mid & g_2^{g_1} = g_2 g_4,\ g_2^{g_1^{-1}} = g_2 g_4^{-1}, \\ & g_3^{g_1} = g_3 g_5,\ g_3^{g_1^{-1}} = g_3 g_5^{-1}, \\ & g_3^{g_2} = g_3 g_6,\ g_3^{g_2^{-1}} = g_3 g_6^{-1}, \\ & g_j^{g_i^{\pm 1}} = g_j \text{ for all other } i < j \rangle \end{aligned}$$

which has the abelian segmentation $\{\{1,2\},\ \{3,4,5,6\}\}$ and computed canonical induced sequences for random subgroups U and N with N normalized by U. We then proceeded to compute the intersection using both algorithm 3.2 as well as the algorithm of [Eic00]. Each computation was repeated one hundred times to account for fluctuations. Table 3 contains the total runtime in milliseconds for some of these tests as well as the corresponding generator matrices.

U	N	[Eic00] (in ms)	3.2 (in ms)
$\begin{pmatrix} 1&4&0&4&0&2\\ &6&0&0&0&2\\ &&1&4&0&0\\ &&&6&0&0\\ &&&&1&4\\ &&&&&6 \end{pmatrix}$	$\begin{pmatrix} 1&0&0&0&0&0\\ &2&0&0&0&1\\ &&1&1&0&0\\ &&&2&0&0\\ &&&&1&0\\ &&&&&2 \end{pmatrix}$	660	612
$\begin{pmatrix} 1&0&14&0&6&18\\ &3&4&1&3&9\\ &&16&1&10&20\\ &&&3&4&6\\ &&&&16&0\\ &&&&&48 \end{pmatrix}$	$\begin{pmatrix} 1&0&0&2&0&1\\ &3&0&1&0&1\\ &&1&2&0&1\\ &&&3&0&0\\ &&&&1&0\\ &&&&&3 \end{pmatrix}$	729	792
$\begin{pmatrix} 3&1&1&5&3&3\\ &2&1&3&0&0\\ &&2&5&4&0\\ &&&6&3&3\\ &&&&6&2\\ &&&&&4 \end{pmatrix}$	$\begin{pmatrix} 3&0&0&2&1&0\\ &2&0&4&1&0\\ &&1&2&1&0\\ &&&3&0&0\\ &&&&3&0\\ &&&&&1 \end{pmatrix}$	696	716

Table 3: Runtime comparison of intersection algorithms

The data suggests that there isn't a clear answer which algorithm to prefer. However, this changes as soon as we replace the canonical induced sequence by arbitrary generating sets. Table 4 shows the results in this case.

U	N	[Eic00] (in ms)	3.2 (in ms)
$\begin{pmatrix} 2&2&0&3&0&-2\\ 2&1&1&-2&-1&2\\ 4&2&0&2&0&0\\ 0&-1&-1&-1&-2&0 \end{pmatrix}$	$\begin{pmatrix} -1&0&3&-5&5&-3\\ 0&-5&-3&1&3&3\\ 1&-1&-2&-4&-3&1\\ 5&2&1&0&1&2 \end{pmatrix}$	938	1802
$\begin{pmatrix} 0&1&-3&5&2&1\\ -2&-3&2&-4&0&0\\ 0&1&-1&1&-4&-2\\ -2&-4&-1&1&1&-2 \end{pmatrix}$	$\begin{pmatrix} 0&-1&-7&-1&0&0\\ -2&3&2&1&-3&-3\\ 0&-4&1&0&-2&-5\\ 0&0&2&2&0&-1 \end{pmatrix}$	920	2017

Table 4: Runtime comparison of intersection algorithms with arbitrary generator sets

Both algorithms suffer a loss in efficiency but the effect on algorithm 3.2 is more substantial. Since the generator sets don't form an induced sequence, the Zassenhaus algorithm has additional work when computing one as it does so in $G \times G$ which has an abelian segmentation of length 3. The algorithm in [Eic00] computes an induced sequence of UN first which is done in G where there is a shorter segmentation. This preprocessing allows for a faster computation in the direct product $U \times UN/N$. Note: In the cases shown in table 4 the group $U \times UN/N$ also has an abelian segmentation of only length 2, additionally simplifying the subsequent computations.

In conclusion the algorithm presented in [Eic00] performs on arbitrary generator sets better than algorithm 3.2 and they tie when executed on canonical induced sequences. Interesting about this is that both algorithms appear to

profit differently from the integration of the new induced sequence algorithm 2.33. This suggests that in some cases the integration of the new approach will make it necessary to revisit older algorithms and decisions in their design to make full use of the improvements. For the intersection algorithm described here as well as the general version in [Eic00] we have provided an implementation in GAP. It will be used later in section 4.5 to compute the core of certain subgroups of polycyclic groups.

4. Residual Nilpotence

This chapter describes methods to determine whether certain types of polycyclic groups are residually nilpotent.

In the first section of this chapter we briefly introduce the notion of residual nilpotent groups and collect some basic properties which we need later on. For a more detailed overview we refer to [CMZ17] and [Seg83]. If the reader is already familiar with the topic, it suffices to skim through this section and return to it once some notation or reference in later sections remains unclear.

Afterwards in section 2 we show that deciding whether a polycyclic abelian-by-nilpotent group is residually nilpotent can algorithmically be reduced to the torsion-free subcase (see corollary 4.12) and using a theorem by Ostheimer [Ost99] to the case where the quotient acts as a nilpotent abelian-by-finite unimodular matrix group on the free-abelian normal subgroup. We then introduce a criterion for testing residual nilpotency based on [Rob82] which allows us to reduce the question to an eigenvalue problem over some matrix group.

In the third and fourth section of this chapter we focus on the cases where the quotient is either finite nilpotent or infinite abelian. For the former we show an analogon to a theorem by [AF11] for abelian-by-finite-cyclic groups which links the residual nilpotency to the order of the occurring eigenvalues (see proposition 4.24). This result is then used to show that a polycyclic free-abelian-by-finite-p group is always residual nilpotent (theorem 4.29) and to construct a necessary condition for a general free-abelian-by-finite-nilpotent group to be residual nilpotent (proposition 4.32). Combining both results yields our algorithm 4.35 which only uses linear algebra for testing residual nilpotency in that case, making it very efficient. In the case where the quotient is infinite abelian we exploit that abelian matrix groups are triagonalizable over the complex numbers. By doing so one can translate the eigenvalue criterion to an ideal membership problem in an order of an algebraic number field. This is described in algorithm 4.41.

We close off this chapter by describing an algorithm for finding a residual nilpotent subgroup of finite index in an arbitrary polycyclic group which is

based on a proof by Segal and uses the methods described in the previous chapters of this work. Unfortunately, this algorithm cannot be used to check for residual nilpotency itself as it does not construct a maximal residually nilpotent subgroup.

4.1. Definitions And Basic Results

4.1. Definition
For a group G and $A, B \subseteq G$ we write

$$[A, B] := \langle a^{-1}b^{-1}ab \mid a \in A,\ b \in B \rangle$$

for the subgroup generated by the commutators of A and B. For $n \in \mathbb{N}_{>1}$ we define recursively

$$[A, {}_nB] := [[A, {}_{n-1}B], B] \text{ with } [A, {}_1B] := [A, B].$$

Note that with this notation we have that the commutator subgroup of G is $[G, G]$ and the *lower central series* of G is

$$G \geq [G, G] \geq [G, {}_2G] \geq [G, {}_3G] \geq \dots$$

To comply with the commonly used notation, we will also write $\gamma_{n+1}(G)$ for $[G, {}_nG]$ and $\gamma_1(G) := G$. Dual to the lower central series one can define the upper central series

$$1 = \zeta_0(G) \leq \zeta_1(G) \leq \zeta_2(G) \leq \dots$$

by setting $\zeta_{i+1}(G)$ to the canonical lift of the centre of $G/\zeta_i(G)$ in G.

4.2. Definition
A group G is called *nilpotent* if there is an $n \in \mathbb{N}$ with $\gamma_{n+1}(G) = 1$. The smallest n with $\gamma_{n+1}(G) = 1$ is then called the *nilpotency class* of G. We call G *residually nilpotent* if

$$\gamma_\omega(G) := \bigcap_{n=0}^{\infty} \gamma_{n+1}(G) = 1.$$

4.3. Lemma
A group G is nilpotent with $\gamma_{c+1}(G) = 1$ if and only if $\zeta_c(G) = G$.

The statement and its proof can be found for example in [Seg83, ch. 1.B, prop. 7, p. 6].

4.4. Remark
A nilpotent group is of course residually nilpotent. Conversely a finite residually nilpotent group is also nilpotent as the lower central series has to become stationary.

We state a few basic properties of (residually) nilpotent groups, which we will use in the following discussion.

4.5. Lemma
Let G be a group, $U,\ V$ subgroups of G and N a normal subgroup of G.

1. If G is (residually) nilpotent then U is (residually) nilpotent.
2. For all i we have $\gamma_i(G/N) = \gamma_i(G)N/N$.
3. If G is nilpotent, then G/N is nilpotent.
4. A finitely generated nilpotent group is polycyclic.
5. The set $\tau(G)$ of all elements of finite order in a nilpotent group G forms a characteristic subgroup of G. If G is finitely generated, $\tau(G)$ is finite.

Proof.

1. Follows directly from $\gamma_i(U) \leq \gamma_i(G)$.
2. See for example [CMZ17, Cor. 2.1].
3. Follows directly from 2.
4. The idea of the proof is to refine a central series of G to a polycyclic one. See for instance [CMZ17, Thm. 4.4].
5. The proof can be found in [Seg83, ch. 1.B, cor. 10, p. 13]. □

The following lemma is a direct consequence of lemma 4.5. We will use it later when considering the action of a group on a normal subgroup.

4.6. Lemma
Let G be a group and N a normal subgroup such that G/N is nilpotent. Then G is residually nilpotent if and only if

$$\bigcap_{n=1}^{\infty} [N, {}_nG] = 1.$$

Proof. As $[N, {}_nG] \subseteq [G, {}_nG]$ the "only if" part is trivial. For the converse observe that since G/N is nilpotent there is a c such that $\gamma_{c+1}(G) \cdot N/N = \gamma_{c+1}(G/N) = 1$ by lemma 4.5. Hence, $\gamma_{c+1}(G) \subseteq N$ and the claim follows from

$$\bigcap_{n=1}^{\infty} \gamma_{n+1}(G) = \bigcap_{n=c+1}^{\infty} \gamma_{n+1}(G) = \bigcap_{n=1}^{\infty} [\gamma_{c+1}(G), {}_nG] \subseteq \bigcap_{n=1}^{\infty} [N, {}_nG] = 1. \qquad \square$$

4.7. Remark
In contrast to lemma 4.5 (3) we have that quotient groups of a residually nilpotent group G do not need to be residually nilpotent even though the equality of part (2) holds. An example is given by the group

$$G = \left\langle \begin{pmatrix} -31 & 9 \\ 7 & -2 \end{pmatrix} \right\rangle \ltimes \mathbb{Z}^2.$$

This group is residually nilpotent, which for instance can be verified by using proposition 4.24. Denote by g_2, g_3 the standard basis of $\mathbb{Z}^2$. If we choose the normal subgroup $N = \langle g_2^2,\ g_3^2 \rangle$, we get

$$\gamma_i(G/N) = \langle g_2 N,\ g_3 N \rangle \text{ for all } i.$$

Thus G/N is not residually nilpotent.

In certain cases quotients of residually nilpotent groups are residually nilpotent themselves. We will make use of the criterion below.

4.8. Proposition
Let G be a residually nilpotent group and N a normal subgroup of G. Assume there is a $k > 0$ such that $|\gamma_k(G) \cap N|$ is finite, then G/N is residually nilpotent.

Proof. Assume G/N is not residually nilpotent. Then we have a $g \in G \setminus N$ with $gN \in \gamma_i(G/N)$ for all i. Using lemma 4.5 (2) we get that there is a $g_i \in \gamma_i(G)$ with $g_i N = gN$ for all i. For all $i \geq k$ we have $g_i^{-1} g_k \in \gamma_k(G) \cap N$. As this intersection is finite there are only finitely many different g_i. But G is residually nilpotent so there is an $l > k$ such that none of the g_i is an element of $\gamma_l(G)$ which yields a contradiction. $\square$

4.2. Abelian-By-Nilpotent Groups

We begin with a theorem by Stroud and Lennox-Roseblade. For a proof see [Rob82, 15.3.6 and 15.3.7].

4.9. Proposition
Let G be a finitely generated group with an abelian normal subgroup N such that G/N is nilpotent.

1. There is an integer n such that for every subgroup $U \leq G$ with UN normal in G we have
$$\gamma_n(U) \cap Z(U) = 1.$$
2. For every normal subgroup U of G we have $\gamma_\omega(U) = [\gamma_\omega(U), U]$.

4.10. Corollary
Let G be a finitely generated abelian-by-nilpotent group. Then G is residually nilpotent if and only if $G/Z(G)$ is residually nilpotent.

Proof. Let G be residually nilpotent. Using proposition 4.9 we get that the centre of G fulfils the conditions of proposition 4.8 and $G/Z(G)$ is residually nilpotent. Conversely, if $G/Z(G)$ is residually nilpotent, we have $\gamma_\omega(G) \leq Z(G)$. Again by proposition 4.9 there is an integer n such that the intersection $\gamma_n(G) \cap Z(G)$ is trivial and we get that $\gamma_\omega(G) = 1$. So G is residually nilpotent. □

An immediate consequence of corollary 4.10 is that for polycyclic abelian-by-nilpotent groups we can restrict ourselves to centreless groups.
Another application of proposition 4.9 is the following proposition. Recall that a group is called *characteristically simple* if it has no proper non-trivial characteristic subgroup.

4.11. Proposition
Let G be a finitely generated group with a finitely generated normal abelian subgroup N such that G/N is nilpotent. Denote by U a finite characteristically simple normal subgroup of G. Then G is residually nilpotent if and only if G/U is residually nilpotent and $[U, G] = 1$.

Proof. Without loss of generality assume $U \neq 1$. Assume that G is residually nilpotent. The quotient group G/U is residually nilpotent by 4.8 as U is finite. The subgroup U is characteristically simple, therefore we have either $[U, G] = 1$ or $[U, G] = U$. Hence U has to be trivial as G is residually nilpotent. For the converse we apply 4.9: As G/U is residually nilpotent we have $\gamma_\omega(G) \leq U$ which is characteristic in U and by 4.9 we have $\gamma_\omega(G) = [\gamma_\omega(G), G]$. So by our assumption $\gamma_\omega(G) = 1$ and G is residually nilpotent. □

Note that deciding whether a polycyclic abelian-by-nilpotent group is residually nilpotent therefore reduces to deciding whether a free-abelian-by-nilpotent group is residually nilpotent:

4.12. Corollary
Let G be a polycyclic group with a normal abelian subgroup N such that G/N is nilpotent. Denote by $\tau(N)$ the torsion subgroup of N and assume we have a series

$$\tau(N) = U_1 \geq U_2 \geq \ldots \geq U_{k+1} = 1$$

of normal subgroups of G such that the quotients U_i/U_{i+1} are elementary abelian (and non-trivial). Then G is residually nilpotent if and only if $G/\tau(G)$ is residually nilpotent and for all $i = 1, \ldots, k$ we have $[U_i, G] \leq U_{i+1}$.

Proof. If G is residually nilpotent, we have that all G/U_i are residually nilpotent as the U_i are all finite and we can apply lemma 4.8. For $i = 1$ this shows that $G/\tau(N)$ is residually nilpotent. As G/U_{i+1} is residually nilpotent we have that $\gamma_\omega(G/U_{i+1}) \leq U_i/U_{i+1}$. The latter is elementary abelian and therefore characteristically simple and we can apply proposition 4.11 to get that $[U_i/U_{i+1}, G/U_{i+1}] = 1$. Now $[U_i, G] \leq U_{i+1}$ follows. For the converse one applies proposition 4.11 inductively to show that the quotient groups G/U_i are residually nilpotent for all i analogously to the argument above. □

Computing such a series for the torsion subgroup of N can be done by using the structure theorem of finite abelian groups and refining a series given by the Sylow subgroups. A more sophisticated approach is discussed in [Eic00, ch. 8.2].

The following definitions provide the necessary tools to establish the main criterion which we use to test abelian-by-nilpotent groups on residual nilpotency. The idea is to translate lemma 4.6 into module theory: For $N \trianglelefteq G$ with N abelian and G/N nilpotent the conjugation defines an action of G/N on N. The commutator group $[N, G]$ is generated by all the $[n, g] = n^{-1}n^g$ with $n \in N,\ g \in G$. Therefore, we can identify $[N, {}_kG]$ as the right submodule $N \cdot I^k_{G/N}$ where $I^k_{G/N} = \{a_1 \cdot \ldots \cdot a_k \mid a_i \in I_{G/N}\}$ and $I_{G/N}$ is defined the following way:

4.13. Definition
Let G be a group. Then we call the (additive) subgroup I_G generated by all the $g - 1 \in \mathbb{Z}[G]$ for $g \in G$ the augmentation (right) ideal of $\mathbb{Z}[G]$.

I_G is free with basis $\{g - 1 \mid g \in G \setminus \{1\}\}$. As the name already suggests, one can show that I_G is indeed a (right) ideal in the group ring $\mathbb{Z}[G]$.

For a result by Robinson we need the following definition.

4.14. Definition
Let R be a ring with identity element (not necessarily commutative). We say that an element x is *central* if it commutes with every element of R: $xr = rx$ for all $r \in R$. An ideal I of R is called *central* if it is generated by a set of

central elements in R. An ideal I is called a *polycentral ideal* if there is a finite series of ideals of R

$$0 = I_0 < I_1 < \ldots < I_k = I$$

such that every I_{i+1}/I_i is a central ideal of R/I_i.

We cite the following proposition by Robinson:

4.15. Proposition ([Rob82, 15.3.8])
Let R be a ring with identity element, I a polycentral ideal and M a right Noetherian R-module. Then an element $a \in M$ belongs to $M \cdot I^n$ for every $n > 0$ if and only if $a = ax$ for some $x \in I$.

4.16. Remark
Let G be a nilpotent group. By [Rob82] the augmentation ideal I_G of $\mathbb{Z}[G]$ is polycentral: Let

$$1 = \zeta_0(G) \trianglelefteq \ldots \trianglelefteq \zeta_c(G) = G$$

be the upper central series. One then defines the right ideals I_i generated by all $x - 1$ where $x \in \zeta_i(G)$ and shows that the I_i are in fact two-sided ideals. Also the quotients I_{i+1}/I_i are central ideals in R/I_i. Choose $x \in \zeta_{i+1}(G)$ and $g \in G$, then we have

$$(x-1)g - g(x-1) = xg - gx = gx([x,g] - 1).$$

As $[x, g] \in \zeta_i(G)$ the right hand side is 0 in R/I_i and $x - 1$ is a central element in the quotient ring.

Putting lemma 4.6 and proposition 4.15 together we immediately get the following criterion to decide whether a finitely generated abelian-by-nilpotent group is residually nilpotent.

4.17. Corollary
Let G be a finitely generated group and N a normal abelian subgroup such that G/N is nilpotent. Then G is not residually nilpotent if and only if there is an $x \in I_{G/N}$ and an $a \in N \setminus \{0\}$ such that $a = ax$.

For the rest of this chapter we will only consider polycyclic free-abelian-by-nilpotent groups as outlined by corollary 4.12. So assume we have a polycyclic group G with normal subgroup $N \cong \mathbb{Z}^n$ such that G/N is nilpotent. Also without loss of generality we may assume that $n > 0$ and consider the representation

$$\varphi : G/N \to \mathrm{GL}_n(\mathbb{Z})$$

given by the action of G/N on N. Simplifying notation, we will also write $\varphi : \mathbb{Z}[G/N] \to \mathrm{Mat}(n, \mathbb{Z})$ for the corresponding ring homomorphism turning N

into a $\mathbb{Z}[G/N]$-module. Using this representation we can now rephrase corollary 4.17 into an eigenvalue problem: G is not residually nilpotent if and only if there is $x \in I_{G/N}$ such that $\varphi(x)$ has a non-trivial eigenvector $a \in N \cong \mathbb{Z}^n$ to the eigenvalue 1. The advantage of this approach is that in order to decide whether G is residually nilpotent, it suffices to analyse only the image of the representation in $\mathrm{GL}_n(\mathbb{Z})$.

4.18. Lemma
Let G be a finitely generated group and $N \cong \mathbb{Z}^n$ a free abelian normal subgroup such that G/N is nilpotent. As above denote by

$$\varphi : G/N \to \mathrm{GL}_n(\mathbb{Z})$$

the representation given by the action of G/N on N. Then G is residually nilpotent if and only if $\operatorname{Im}\varphi \ltimes_\varphi N$ is residually nilpotent.

Proof. This follows immediately from 4.17 as the representation

$$\varphi' : \operatorname{Im}\varphi \ltimes_\varphi N \to \mathrm{GL}_n(\mathbb{Z})$$

fulfils $\varphi = \varphi' \circ e$ where e is the canonical embedding. So the images of the respective augmentation ideals coincide. □

Note that the following equivalence holds:

4.19. Lemma
Let $M \in \mathrm{Mat}(n \times n, \mathbb{Q})$ and $\lambda \in \mathbb{Q}$. Then the following statements are equivalent:

1. There is a vector $v \in \mathbb{Z}^n \setminus \{0\}$ such that $vM = \lambda v$.
2. There is a vector $v \in \mathbb{Q}^n \setminus \{0\}$ such that $vM = \lambda v$.
3. There is a vector $v \in \mathbb{C}^n \setminus \{0\}$ such that $vM = \lambda v$.

Proof. The equivalence of 1 and 2 is trivial as one can scale any eigenvector in $\mathbb{Q}^n$ by the common denominator. The equivalence of 2 and 3 follows as the existence of an eigenvector in $\mathbb{C}^n$ implies that $\mathrm{rk}(M - \lambda \cdot E_n) < n$. As $M - \lambda \cdot E_n$ is in $\mathrm{Mat}(n \times n, \mathbb{Q})$ there must be a non-trivial rational vector in its kernel. □

The implication of this is that as we are only concerned about the existence of an eigenvector, we don't need to worry about whether it is contained in $\mathbb{Z}^n$ when using techniques such as changing bases in a suitable larger field. This allows us to decompose the representation taking a K-irreducible subspace and block triangularizing the representation with a basis change:

4.20. Lemma
Let K be a field and V a finite dimensional K-vector space. Let G be a group and $\varphi : G \to \mathrm{GL}(V)$ a representation of G. Assume that there is an irreducible subspace U. Denote by $\psi : G \to \mathrm{GL}(V/U)$ the representation on V/U given by a basis extension of U and choose a $\lambda \in K$. Then for all $g \in G$ we have

$$\mathrm{Eig}(\varphi(g), \lambda) \neq 0 \iff (\mathrm{Eig}(\varphi(g)|_U, \lambda) \neq 0 \text{ or } \mathrm{Eig}(\psi(g), \lambda) \neq 0)\,.$$

Proof. By choosing a basis of V which extends one of U the images of the representation $\varphi : \mathbb{Z}[G] \to \mathrm{Mat}(n, K)$ will be given by block triangular matrices of the form

$$\begin{pmatrix} A & B \\ 0 & C \end{pmatrix}$$

where the upper left block corresponds to $\varphi|_U$ and the lower right to ψ. So the claim just follows from the fact that the characteristic polynomial of such a matrix splits into the product of the characteristic polynomials of A and C respectively. □

As we are only interested in the polycyclic setting we can use a theorem by Ostheimer about the structure of polycyclic matrix groups.

4.21. Proposition ([Ost99])
Let G be a polycyclic-by-finite subgroup of $\mathrm{GL}(n, \mathbb{Z})$. Then there is a basis of $\mathbb{Z}^n$ such that all matrices g in G are in block triangular form

$$g = \begin{pmatrix} a_1 & * & * & * \\ 0 & a_2 & * & * \\ 0 & 0 & \ddots & * \\ 0 & \dots & 0 & a_r \end{pmatrix}$$

where the image of G under the map that takes g to a_i is abelian-by-finite.

This theorem shows that deciding whether a finitely generated free-abelian-by-nilpotent group is residually nilpotent can be reduced to the case of free-abelian-by-(abelian-by-finite and nilpotent) groups. In this thesis we are addressing the case where the quotient is an abelian or a finite nilpotent group which will be the subject of the following discussion. The advantage of this is that the structure of the matrix groups are even simpler: Finite matrix groups are simultaneously diagonalizable over the complex numbers and abelian ones can at least be triangularized. This makes it easier to solve the eigenvalue problem as it boils down to a discussion about the diagonal elements.

4.3. Free-Abelian-By-Finite-Nilpotent

In this section, we first consider the case where G is a finitely generated free-abelian-by-finite-nilpotent group. We are going to give a constructive criterion how to decide whether G is residually nilpotent. Afterwards we state an algorithm for this which only relies on rational linear algebra. In order to do this we first prove the statement for the case where $N \cong \mathbb{Z}^n$ is a normal subgroup such that G/N is cyclic and show that the general case can be reduced to this. The case $G/N \cong \mathbb{Z}$ was already covered by [AF11, 2.7] where the authors showed:

4.22. Lemma
Let ψ be an automorphism of $\mathbb{Z}^n$ and write χ as the characteristic polynomial of ψ. Then $G = \mathbb{Z}^n \rtimes_\psi \mathbb{Z}$ is residually nilpotent if and only if $\chi_i(1) \neq \pm 1$ for each irreducible factor χ_i of $\chi \in \mathbb{Z}[t]$.

We will prove this statement also for the case where G/N is a finite cyclic group. Recall that we have a representation

$$\varphi : G/N \to \mathrm{GL}(n, \mathbb{Z})$$

and if g denotes a generator of G/N, $\varphi(g)$ generates the image and is an invertible integer matrix of finite order. Therefore $\varphi(g)$ has only roots of unity as eigenvalues. Hence the irreducible factors of the characteristic polynomial are cyclotomic polynomials. Their values in 1 are commonly known see for instance [Sch84, 22.8]:

4.23. Lemma
Let ζ be an m-th root of unity ($m > 1$) and Φ_m its minimal polynomial over $\mathbb{Z}$. Then the following holds

$$\Phi_m(1) = \begin{cases} p & \text{if } m = p^k \text{ for } p \text{ a prime number} \\ 1 & \text{else.} \end{cases}$$

One can show this by induction on the recursive formula of the cyclotomic polynomial

$$\Phi_n(x) = \frac{x^n - 1}{\prod_{\substack{d|n \\ d<n}} \Phi_d(x)}.$$

With this we can rephrase 4.22 in the finite case as

4.24. Proposition
Let G be a group with a normal subgroup $N \cong \mathbb{Z}^n$ such that G/N is a finite cyclic group and denote by g a generator of G/N. Then G is residually nilpotent if and only if the eigenvalues of $\varphi(g)$ have only prime power order.

Before proving this we note this immediate corollary.

4.25. Corollary
Let G be a group with a normal subgroup $N \cong \mathbb{Z}^n$ such that $G/N \cong \mathbb{Z}/p^k\mathbb{Z}$ for a prime power p^k. Then G is residually nilpotent.

4.26. Example
Note that 4.25 is only sufficient but not necessary. We define

$$A = \begin{pmatrix} -1 & & & \\ & -1 & & \\ & & 0 & 1 \\ & & -1 & -1 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 1 & & \\ -1 & 0 & & \\ & & 1 & 1 \\ & & -1 & 0 \end{pmatrix} \in \mathrm{GL}(4,\mathbb{Z}).$$

As $\mathrm{ord}(A) = \mathrm{ord}(B) = 6$ we have that $\langle A\rangle \cong \mathbb{Z}/6\mathbb{Z} \cong \langle B\rangle$. If we now look at the semidirect product $G = \mathbb{Z}^4 \rtimes \langle A\rangle$, then the image of the augmentation ideal under φ is generated by $A - \mathrm{Id}$. $(A-\mathrm{Id})^6 = \mathrm{diag}(64, 64, -27, -27)$ and as the group ring is commutative $(A-\mathrm{Id})^6$ divides $\varphi(x)^6$ for all x of the augmentation ideal. Hence, no element of the augmentation ideal has a non-zero fixvector and G is residually nilpotent.
In contrast for $G = \mathbb{Z}^4 \rtimes \langle B\rangle$ we have $(B - \mathrm{Id})^3 = \mathrm{Id}$ is an element of the image of the augmentation ideal. This shows that knowing the isomorphism type of G/N and N is not enough to decide whether G is residually nilpotent. It explicitly depends on the action on the normal subgroup.

An alternative proof of 4.25 can be derived from the following lemma. We state it here as we will need it to prove the more general theorem 4.29.

4.27. Lemma
Let $g \in \mathrm{GL}(n,\mathbb{Z})$ be an element of finite order. If $\mathrm{ord}(g) = p^k$ for some prime p and $k > 0$ then

$$\mathbb{Z}^n(g-1)^{p^k} \subseteq p\mathbb{Z}^n.$$

Proof. This follows from binomial expansion:

$$(g-1)^{p^k} = \sum_{i=0}^{p^k} \binom{p^k}{i} g^i(-1)^{p^k-i} = (g^{p^k} + (-1)^{p^k}) + \sum_{i=1}^{p^k-1} \binom{p^k}{i} g^i(-1)^{p^k-i}.$$

The rightmost sum is divisible by p because all of the binomial coefficients are as p^k is a prime power and $1 \le i < p^k$. On the other hand $g^{p^k} + (-1)^{p^k} = 1 + (-1)^{p^k}$ is zero for odd primes p and 2 for $p = 2$. So the entire right hand side is divisible by p and the claim follows. □

From this we get 4.25: Denote by g the generator of the quotient group G/N. Then the augmentation ideal $I_{G/N}$ is generated by $g-1$ and its m-th power by $(g-1)^m$. Recall from the introduction of the augmentation ideal in definition 4.13 that by switching from additive to multiplicative notation we can identify $[N, {}_{p^k}G]$ as $N \cdot (I_{G/N})^{p^k}$. So by lemma 4.27 using $N \cong \mathbb{Z}^n$ it follows that

$$[N, {}_{p^k}G] = N \cdot (I_{G/N})^{p^k} \subseteq pN.$$

Hence for all $l > 0$ there is an $m > 0$ such that

$$[N, {}_{m}G] \subseteq p^l N$$

and from 4.6 we get

$$\bigcap_{j=1}^{\infty} [N, {}_{j}G] \subseteq \bigcap_{j=1}^{\infty} p^j N = 1$$

and G is residually nilpotent.

Now back to the

Proof of 4.24. We use 4.17 for the proof. As G/N is cyclic we can view $\mathbb{Z}[G/N]$ as the image of

$$\mathbb{Z}[t] \to \mathbb{Z}[G/N], \quad t \mapsto g.$$

The augmentation ideal $I_{G/N}$ is generated by $g-1$, thus its preimage under this map is

$$\{f \in \mathbb{Z}[t] \mid f(1) = 0\}.$$

As G/N is finite (and abelian), the image of g under the representation φ is diagonalizable over the complex numbers and there is a basis of $\mathbb{C}^n$ consisting of eigenvectors of $\varphi(g)$. Let $\lambda \in \mathbb{C}$ be an eigenvalue of $\varphi(g)$ and $v \in \mathbb{C}^n$ a corresponding eigenvector. Then for every $f \in \mathbb{Z}[t]$ the vector v will also be an eigenvector of $\varphi(f(g))$ to the eigenvalue $f(\lambda)$ and conversely every eigenvalue of $\varphi(f(g))$ will be of this form. Putting everything together, 4.17 tells us that G is not residually nilpotent if and only if there is an eigenvalue λ of $\varphi(g)$ and a polynomial $f \in \mathbb{Z}[t]$ such that

$$f(\lambda) = 1 \quad \text{and} \quad f(1) = 0$$

or equivalently (by subtracting 1) if and only if there is an eigenvalue λ of $\varphi(g)$ and a polynomial $f \in \mathbb{Z}[t]$ such that

$$f(\lambda) = 0 \quad \text{and} \quad f(1) = -1.$$

As λ is a primitive m-th root of unity, Φ_m has to divide such an f by the first condition. This on the other hand implies that by the second condition $\Phi_m(1)$ divides $f(1) = -1$, hence $|\Phi_m(1)| = 1$, which by lemma 4.23 means that m cannot be a prime power. Conversely, if m is not a prime power $-\Phi_m$ satisfies these properties by 4.23. □

4.28. Remark
One can extend proposition 4.24 to free-abelian-by-finite-abelian groups. As we are going to prove a stronger result in theorem 4.34 we only sketch the necessary adaptations to the proof: Let G be a group and $N \cong \mathbb{Z}^n$ a free normal subgroup such that G/N is finite abelian. As usual write φ for the representation given by the action of G/N on N. By the theorem of Maschke (see e.g. [Ser77, Theorem 1]) there is a basis $u_1, \ldots, u_n \in \mathbb{C}^n$ such that the $U_i = \langle u_i \rangle$ are $\varphi(g)$ invariant for all $g \in G/N$ and the representation φ can be diagonalized over the complex numbers. By Lemma 4.19 the augmentation ideal of $\operatorname{Im}\varphi$ contains an element with eigenvalue 1 if and only if it does after the change to this basis. The subrepresentations have cyclic image and we can argue as in the proof of 4.24 that they are all cyclic p-groups if and only if G is residually nilpotent. As a consequence we obtain that a free-abelian-by-finite-abelian subgroup is not residually nilpotent if and only if it contains a free-abelian-by-finite-cyclic subgroup which is not residually nilpotent.

Now we want to generalize this idea for all free-abelian-by-finite-nilpotent groups. For this, we proceed in two steps. First we show the equivalent statement of corollary 4.25 in theorem 4.29 below. Afterwards we develop the necessary tools to deal with the case as in example 4.26.

4.29. Theorem
Let G be a group and $N \cong \mathbb{Z}^n$ a free normal subgroup of rank $n > 0$ such that G/N is a finite nilpotent group. As usual write $\varphi : G/N \to \mathrm{GL}(n, \mathbb{Z})$ for the representation given by the conjugation. If $\operatorname{Im}\varphi$ is a finite p-group for a prime p, then G is residually nilpotent. In particular, if G/N is a p-group, then G is residually nilpotent.

Proof. We prove by induction over the polycentral sequence

$$0 = I_0 < I_1 < \ldots < I_k = I_{\operatorname{Im}\varphi}$$

of the augmentation ideal that for each $j = 1, \ldots, k$ there is a natural number $l \in \mathbb{Z}_{>0}$ with $N \cdot g^l \subseteq p \cdot N$ for all $g \in I_j$. The induction start is trivial as $N \cdot 0^1 = 0 \subseteq pN$. Let's assume there is a number l such that $N \cdot g^l \subseteq p \cdot N$ for all $g \in I_j$ and choose $h \in I_{j+1}$. As G/N is finite and nilpotent we can write

$$I_{j+1} = I(1) + \ldots + I(r)$$

where each of the $I(i)$ is the ideal generated by some $x_i - 1 \in I_{j+1}$ in $\mathbb{Z}[\operatorname{Im}\varphi]$. We set $q = |\operatorname{Im}\varphi|$, $l' = r(q-1)+1$ and write $h = h_1 + \ldots + h_r$. Then

$$h^{l'} =: \sum_t \tilde{h}_t$$

is the sum over the r^l many products $\tilde{h}_t$ of l many of the h_s. In each $\tilde{h}_t$ there occurs at least one of the h_s at least q times. As the elements of I_{j+1} lie in the centre of R/I_j, we can write

$$\begin{aligned} h^{l'} &= \sum_t \tilde{h}_t = \sum_t \big((x_{s_t}-1)^q \cdot y_t + a_t\big) && \text{for certain } y_t \in \mathbb{Z}[\operatorname{Im}\varphi],\ a_t \in I_j \\ &= \sum_t (x_{s_t}-1)^q \cdot y_t + A && \text{for certain } y_t \in \mathbb{Z}[\operatorname{Im}\varphi],\ A \in I_j \end{aligned}$$

We already showed in 4.27 that for all $x \in \operatorname{Im}\varphi$: $N(x-1)^q \subset pN$. Hence, we conclude $Nh^{l'} \subset pN + N \cdot A$. So by taking the power $(h^{l'})^l$ each summand contains one of the $(x-1)^q$ except for A^l where we apply the induction hypothesis and conclude that $N(h^{l'})^l \subseteq pN$.

It follows that there is no element in $x \in I_{\operatorname{Im}\varphi}$ such that $ax = a$ for some non-trivial $a \in N$. Applying 4.17 we conclude that G is residually nilpotent. □

4.30. Lemma
Let V be a free R-module and let G be a subgroup of $\mathrm{GL}(V)$. Assume there are $P,\ Q \leq G$ such that $P \times Q = G$. Then $\bigcap_{\psi\in P} \operatorname{Eig}(\psi, 1)$ is a G-invariant subspace of V.

Proof. The intersection $\bigcap_{\psi\in P} \operatorname{Eig}(\psi, 1)$ is certainly invariant under the action of P. Let ψ' be an element of Q. Then we have for all $\psi \in P$ and $v \in \bigcap_{\psi\in P} \operatorname{Eig}(\psi, 1)$

$$\psi(\psi'(v)) = \psi'(\psi(v)) = \psi'(v)$$

as ψ and ψ' commute. Thus $\psi'(v) \in \operatorname{Eig}(\psi, 1)$ for all $\psi \in P$. So $\bigcap_{\psi\in P} \operatorname{Eig}(\psi, 1)$ is also Q-invariant and the claim follows. □

4.31. Lemma
Let $G = P \times Q$ be a subgroup of $\mathrm{GL}(n, \mathbb{Z})$. Then $\mathbb{Z}^n / \bigcap_{\psi\in P} \operatorname{Eig}(\psi, 1)$ is a free $\mathbb{Z}$-module.

Proof. Let $v \in \mathbb{Z}^n$ such that the class of v has finite order k in the quotient $\mathbb{Z}^n / \bigcap_{\psi\in P} \operatorname{Eig}(\psi, 1)$ so $k \cdot v \in \bigcap_{\psi\in P} \operatorname{Eig}(\psi, 1)$. Then for all $\psi \in P$ we have

$$k \cdot \psi(v) = \psi(kv) = kv$$

and $v = \psi(v)$ follows. So $v \in \bigcap_{\psi\in P} \operatorname{Eig}(\psi, 1)$ and $k = 1$. □

4.32. Proposition
Let G be a group and $N \cong \mathbb{Z}^n$ a free normal subgroup of rank $n > 0$ such that G/N is a finite nilpotent group. As usual write

$$\varphi : G/N \to \mathrm{GL}(n, \mathbb{Z}) \hookrightarrow \mathrm{GL}(n, \mathbb{C})$$

for the action of the quotient on N. Suppose there are $P, Q \leq G/N$ with $\gcd(|P|, |Q|) = 1$ where $\varphi(P)$ and $\varphi(Q)$ are non-trivial. If $\bigcap_{f \in P} \operatorname{Eig}(\varphi(f), 1)$ is trivial then there is a subgroup $N \leq U \leq G$ with U/N cyclic which is not residually nilpotent. In particular, G is not residually nilpotent.

Proof. Let $g \in P$ such that $\varphi(g) \neq \mathrm{Id}$. Let $\lambda \neq 1$ be an eigenvalue of g. Since g has finite order, λ is a primitive m-th root of unity for some m. If m is not a prime power we are already done: Setting $U \leq G$ as the preimage of $\varphi^{-1}\langle g \rangle \leq G/N$ under the canonical projection yields the desired subgroup which is not residually nilpotent by 4.24. So assume m is a prime power. Since G/N is a finite nilpotent group it is the direct product of its Sylow p-subgroups. Now $\gcd(|P|, |Q|) = 1$ so every $f \in Q$ commutes with g and the eigenspace decomposes

$$\operatorname{Eig}(\varphi(g), \lambda) = \bigoplus_{\mu \in \operatorname{Spec} \varphi(f)} (\operatorname{Eig}(\varphi(g), \lambda) \cap \operatorname{Eig}(\varphi(f), \mu)) \tag{5}$$

as f is diagonalizable. Assume for all $f \in Q$ and all its eigenvalues $\mu \neq 1$ we have

$$\operatorname{Eig}(\varphi(g), \lambda) \cap \operatorname{Eig}(\varphi(f), \mu) = 0.$$

Then equation (5) implies that

$$0 \neq \operatorname{Eig}(\varphi(g), \lambda) \leq \operatorname{Eig}(\varphi(f), 1) \text{ for all } f \in Q.$$

But this yields a contradiction as $\bigcap_{f \in Q} \operatorname{Eig}(\varphi(f), 1)$ is trivial. So we have an $f \in Q$ with eigenvalue $\mu \neq 1$ a k-th primitive root of unity where k is coprime to m such that

$$\operatorname{Eig}(\varphi(g), \lambda) \cap \operatorname{Eig}(\varphi(f), \mu) \neq 0 \tag{6}$$

Any non-trivial element of the intersection in equation (6) is an eigenvector of $\varphi(fg)$ to the eigenvalue $\lambda\mu$ which is a $k \cdot m$-th primitive root of unity as k and m are coprime. Thus taking the preimage of $\varphi^{-1}\langle f \cdot g \rangle$ under the canonical projection $G \to G/N$ gives the desired group which is again not residually nilpotent by proposition 4.24. □

4.33. Corollary
Let G be a group and $N \cong \mathbb{Z}^n$ a free abelian normal subgroup of rank $n > 0$ such that G/N is a finite nilpotent group. Write

$$\varphi : G/N \to \mathrm{GL}(n, \mathbb{Z}) \hookrightarrow \mathrm{GL}(n, \mathbb{C})$$

for the action of the quotient on N. Assume φ is irreducible. If $\operatorname{Im} \varphi$ is not a p-group, then G is not residually nilpotent.

Proof. If $\operatorname{Im}\varphi$ is not a p-group, we can find $P,\ Q \leq G$ with $\gcd(|P|,|Q|) = 1$, $P \times Q \leq G$ and where $\varphi(P)$ and $\varphi(Q)$ are non-trivial. As shown in 4.30 $\bigcap_{f\in P} \operatorname{Eig}(\varphi(f), 1)$ is a G-invariant subspace. φ is irreducible so the intersection is either N or trivial. The former is not possible as $\varphi(P) \neq \{\mathrm{Id}\}$. So we can apply 4.32 and G is not residually nilpotent. □

Therefore the question whether a free-abelian-by-finite-nilpotent group is residually nilpotent can be answered by only looking at the cyclic subgroups of the quotient group:

4.34. Theorem
Let G be a group and $N \cong \mathbb{Z}^n$ a free normal subgroup of rank $n > 0$ such that G/N is a finite nilpotent group. Then G is residually nilpotent if and only if every subgroup $H \leq G$ with $N \leq H$ and H/N cyclic is residually nilpotent.

Proof. If G is residually nilpotent, every subgroup is residually nilpotent so there is nothing to show. We prove the converse by contraposition. We embed G/N into $\mathrm{GL}(n,\mathbb{C})$ as usual by considering the action of the conjugation map. G/N is finite so we can use Maschke's theorem to decompose this representation into irreducible subrepresentations. For one of these irreducible subrepresentations - say ψ - it must hold $\psi(H/N)$ is not a p-group or G would be residually nilpotent by 4.29. In this case we can apply 4.33 and conclude that G cannot be residually nilpotent by 4.20. □

The proof of theorem 4.34 is constructive and suggests a first algorithm. Given a finitely generated free-abelian-by-finite-nilpotent group G, decompose the canonical representation of G into its irreducible ($\mathbb{C}$-)subrepresentations. Check whether the image of each of these subrepresentations are p-groups. If so the group is residually nilpotent, if not it isn't. However, one can circumvent computing the irreducible subrepresentations and working over the complex numbers entirely by using that

$$\dim \operatorname{Eig}_{\mathbb{Q}}(\psi, 1) = \dim \operatorname{Eig}_{\mathbb{C}}(\psi, 1)$$

for $\psi \in \mathrm{GL}(n,\mathbb{Q}) \leq \mathrm{GL}(n,\mathbb{C})$ (compare to 4.19). As we only need to check whether the intersection of such subspaces is trivial we can actually work with linear algebra over the rationals. The idea then is to try to decompose the action of the finite nilpotent quotient to the action of its Sylow p-subgroups. If this is possible, this means that there is a change of basis such that the representation is in block diagonal form and each Sylow subgroup acts independently from the others. This is described by the following algorithm.

Algorithm 4.35: Testing Residual Nilpotence

Input: a group G with normal subgroup $N \cong \mathbb{Z}^n$ and G/N finite and nilpotent and corresponding representation $\varphi : G/N \to \mathrm{GL}(n, \mathbb{Z})$

Output: whether G is residually nilpotent or not

Initialize $V = N$ and $H = \varphi(G/N)$.
while *H is not a p-group* **do**
  Compute non-trivial p-groups $P_1, \ldots, P_m \leq H$ with $H = P_1 \times \ldots \times P_m$ and $\gcd(|P_i|, |P_j|) = 1$ for $i \neq j$.
  For each i fix a set of generators $f_{i,1}, \ldots, f_{i,r_i}$ of P_i.
  Compute $U_i = \bigcap_{j=1}^{r_i} \mathrm{Eig}(f_{i,j}, 1)$ for all $i = 1, \ldots, m$.
  Compute $U = \bigcap_{j=2}^{m} U_j$.
  if $U_1 = 0$ *or* $U = 0$ **then**
    Return: *G is not residually nilpotent.*
  if $\dim(U_1 + U) < \dim V$ **then**
    Return: *G is not residually nilpotent.*
  Replace V by U_1 and H by the image of $P_2 \times \ldots \times P_m$ in $\mathrm{GL}(U_1)$.
Return: *G is residually nilpotent.*

Proof. Termination: The algorithm terminates as in each iteration H is replaced by a group of smaller order. Thus, H has to be a p-group eventually.

Correctness: First note that we have $U_i = \bigcap_{f \in P_i} \mathrm{Eig}(f, 1)$ as P_i is generated by $f_{i,1}, \ldots, f_{i,r_i}$ and every vector fixed under the action of the generators is fixed by any combination of them. Also, in each iteration the U_i and U are proper subspaces of V as the $P_i \leq \mathrm{GL}(V)$ are non-trivial. As discussed in lemma 4.30 they are H-invariant subspaces of V and we have seen in 4.32 that if any of them is trivial the group cannot be residually nilpotent. In the case where $\dim(U_1 + U) < \dim V$ we can apply Maschke's theorem to get an invariant complement $0 \neq W$ to the $\mathbb{Q}$-space spanned by $U_1 + U$. Denote by $\psi : H \to \mathrm{GL}(W)$ the corresponding subrepresentation. By definition $\psi(P_1) \neq 1$ and $\psi(P_2 \times \ldots \times P_m) \neq 1$ so there is an $f \in P_1$ with $\psi(f) \neq \mathrm{Id}_W$ and thus a $\lambda \neq 1$ with $\mathrm{Eig}(\psi(f), \lambda) \neq 0$. But as

$$\mathrm{Eig}(\psi(f), \lambda) = \bigoplus_{\mu \in \mathrm{Spec}\, \psi(g)} \mathrm{Eig}(\psi(g), \mu) \cap \mathrm{Eig}(\psi(f), \lambda)$$

for all $g \in P_2 \times \ldots \times P_m$ there must be such a g with eigenvalue $\mu \neq 1$ and

$$\mathrm{Eig}(\psi(g), \mu) \cap \mathrm{Eig}(\psi(f), \lambda) \neq \{0\}.$$

Otherwise $0 \neq \mathrm{Eig}(\psi(f), \lambda) \subset \mathrm{Eig}(\psi(g), 1)$ for all $g \in P_2 \times \ldots \times P_m$ which is impossible as by construction of W we have $\bigcap_{g \in P_2 \times \ldots \times P_m} \mathrm{Eig}(\psi(g), 1) = 0$. Hence, we apply 4.34 and know that the group is not residually nilpotent.

If we reach step 12 of the algorithm, we are in the case where $V = U + U_1$. Note this sum does not need to be direct as there might be a non-trivial intersection. We fix a basis of U_1. Using lemma 4.31 we can extend it with elements of U to a basis of $\mathbb{Z}^n$. With respect to this basis the elements of H are in block triangular form. Using lemma 4.20 we get that the eigenvalue problem of corollary 4.17 has a solution if and only if it has for H restricted to the action on U_1 or to the action on $\mathbb{Z}^n/U_1 \cong U/(U \cap U_1)$. The latter is not possible as the action of H on U is the action of P_1 which is a p-group and we can apply theorem 4.29. The action of H on U_1 is given by the action of $P_2 \times \ldots \times P_m$. If $m = 2$ the loop terminates at this point and we apply theorem 4.29. □

The following example shows the algorithm in action.

4.36. Example
We look at the polycyclic group $G := \mathcal{G}_9$. Its complete polycyclic presentation can be found in the appendix. We moved it there for readability reasons and provide here only the necessary information. The group G is given by nine generators $x_1, \ldots, x_9$ and one may verify that $N = \langle x_5, \ldots, x_9 \rangle$, is a free abelian normal subgroup of rank 5 and $G/N \cong D_{2\cdot 4} \times \mathbb{Z}/3\mathbb{Z} \times \mathbb{Z}/5\mathbb{Z}$. However, the image $\operatorname{Im}\varphi$ is isomorphic to $D_{2\cdot 4} \times \mathbb{Z}/3\mathbb{Z}$ and is the matrix group $\langle X_1, X_2, X_3 \rangle$ where

$$X_1 = \begin{pmatrix} 3 & 108 & -60 & 2 & 48 \\ 0 & 1 & 0 & 0 & 0 \\ 4 & 216 & -119 & 4 & 96 \\ -4 & -216 & 120 & -3 & -96 \\ 5 & 270 & -150 & 5 & 121 \end{pmatrix}, \quad X_2 = \begin{pmatrix} 1 & 30 & -18 & 0 & 14 \\ 0 & 1 & 0 & 0 & 0 \\ -4 & -96 & 49 & -4 & -40 \\ 0 & -60 & 36 & 1 & -28 \\ -5 & -120 & 60 & -5 & -49 \end{pmatrix}$$

are the generators of the dihedral group and

$$X_3 = \begin{pmatrix} -1 & -39 & 25 & -1 & -20 \\ 2 & 0 & 0 & 1 & 0 \\ -6 & -53 & 36 & -3 & -28 \\ 2 & 87 & -55 & 2 & 44 \\ -12 & -66 & 45 & -6 & -35 \end{pmatrix}$$

is the generator of the cyclic group of order 3. Computing the subspace U_1 of common fixvectors of X_1 and X_2 we get

$$U_1 = \operatorname{Lin}((0,1,0,0,0),\ (2,0,0,1,0),\ (0,0,-5,0,4))$$

and the subspace U of fixvectors of X_3 is

$$U = \operatorname{Lin}((-7,8,5,0,0),\ (11,6,0,5,0),\ (9,21,0,0,5)).$$

Both are non-trivial and the sum $U_1 + U$ is the entire space $\mathbb{Q}^5$. So we are in the case where we need to look at the action of $\langle X_3 \rangle$ on U_1. As this is a p-group we conclude that G is residually nilpotent.

4.4. Free-Abelian-By-Abelian

Let G be a polycyclic group and N a free abelian normal subgroup. In the last section we dealt with the case where G/N is a finite nilpotent group. As abelian groups are of course also nilpotent these results can be directly applied to finite abelian G/N (see again remark 4.28). The methods discussed there relied on Maschke's theorem and the idea to diagonalize the involved matrix groups over the complex numbers. In this section we are going to generalize this to the case where G/N is an infinite abelian group. This implies that the representation is no longer diagonalizable but as the quotient is abelian it can still be simultaneously triangularized. (See for example [HJ10, Theorem 2.3.3].) By triangularizing the matrix group we can still decide the residual nilpotence of G by solving the eigenvalue problem given by corollary 4.17.

This is enough to give a first sketch of an algorithm. For a generator set $M_1, \dots, M_k$ of $\operatorname{Im}\varphi \leq \mathrm{GL}(n, \mathbb{Z})$ one computes a basis over $\mathbb{C}^n$ such that every M_i is in upper triangular form with respect to this basis. Denote by $\alpha_{i,j}$ the j-th diagonal entry of M_i. Then for any sum or product of the M_i the j-th diagonal entry is just the sum (respectively product) of the $\alpha_{i,j}$. Hence, there is a matrix in the augmentation ideal of $\operatorname{Im}\varphi$ with 1 as the j-th diagonal entry if and only if 1 is contained in the ideal $I = \langle \alpha_{i,j} - 1 \mid i = 1, \dots, k \rangle$ in $\mathbb{Z}[\alpha_{1,j}, \dots, \alpha_{k,j}]$. The $\alpha_{i,j}$ are all algebraic integers as eigenvalues of unimodular matrices, so I is an ideal in the equation order of $\mathbb{Q}[\alpha_{1,j}, \dots, \alpha_{k,j}]$ and therefore free $\mathbb{Z}$-modules of the same rank as the order $\mathbb{Z}[\alpha_{1,j}, \dots, \alpha_{k,j}]$. By computing a basis of I one can check whether this is the entire ring or equivalently whether $1 \in I$.

In fact this procedure can be applied to all abelian-by-nilpotent groups which have a triangularizable image $\varphi(G/N)$. By [Ost99, cor. 2.8] the triangularizable (over the complex numbers) matrix groups in $\mathrm{GL}(n, \mathbb{Z})$ are precisely those that are unipotent-by-abelian. (A unipotent matrix group in this setting is one for which every element has only the eigenvalue 1.) In fact in our setting this can be further restricted:

4.37. Proposition
A finitely generated abelian-by-finite and nilpotent subgroup G of $\mathrm{GL}(n, \mathbb{Z})$ is triangularizable over $\mathbb{C}$ if and only if it is abelian.

For the proof we need the following lemma.

4.38. Lemma
A torsion-free nilpotent group G is abelian-by-finite if and only if it is abelian.

Proof. Let G be abelian-by-finite. Write N for an abelian normal subgroup such that G/N is finite. As G is torsion-free N must be free abelian. Since G is nilpotent there is a basis of N such that the inner automorphism group of G acting on N can be viewed as upper triangular matrices with integer coefficients. (This result can be found in [CMZ17, 6.17].) As these have finite order, G has to act trivially on N and so $N \leq Z(G)$. By the third isomorphism theorem $G/Z(G)$ has to be finite. But if the centre of a nilpotent group is torsion-free, each quotient in the upper central series has to be torsion-free as well (see for instance [Seg83, ch. 1.B cor. 5 p. 12]) which implies that $G/Z(G)$ is trivial. So G is abelian. □

Proof of 4.37. As G is nilpotent and finitely generated, its torsion subgroup $\tau(G)$ is finite (see lemma 4.5) and $G/\tau(G)$ is a torsion-free abelian-by-finite nilpotent group. By lemma 4.38 we get that $G/\tau(G)$ is abelian and the derived subgroup G' is finite. By [Ost99] G is triangularizable if and only if it is unipotent-by-abelian. As G' is finite and consists of unipotent matrices it has to be trivial. Therefore G is abelian. □

The conclusion of proposition 4.37 is that the algorithm sketched in the introduction of this section is applicable to precisely the free-abelian-by-abelian groups. We can therefore assume that the matrix group given by the quotient is abelian and use this to further improve the algorithm. For this we need the following two lemmas.

4.39. Lemma
Let $f \in K[t]$ be a polynomial over some field K and $\psi \in \mathrm{Hom}(V, V)$ be an endomorphism of a K-vector space V. If f is the product of two coprime polynomials $g, h \in K[t]$ then we have

$$\mathrm{Ker}\, f(\psi) = \mathrm{Ker}\, g(\psi) \oplus \mathrm{Ker}\, h(\psi).$$

Proof. The statement with its proof can be found e.g. in [Kow19, 34.4]. □

4.40. Lemma
Let $\psi_1, \ldots, \psi_n \in \mathrm{Hom}(V, V)$ denote a family of commuting endomorphisms of a finite dimensional K-vector space V. Then there are subspaces $U_1, \ldots, U_k \leq V$ invariant under all ψ_i such that $V = U_1 \oplus \ldots \oplus U_k$ and the characteristic polynomial $\chi_{\psi_i|_{U_j}}$ of each ψ_i restricted to any of the U_j is the power of some monic irreducible polynomial over K.

Proof. Without loss of generality assume that the characteristic polynomial χ_{ψ_1} can be factored into two non-trivial coprime polynomials g, h. Then by 4.39

$$\mathrm{Ker}\, g(\psi_1) \oplus \mathrm{Ker}\, h(\psi_1) = \mathrm{Ker}\, \chi_{\psi_1}(\psi_1) = V$$

where the latter equation follows from the Cayley-Hamilton theorem. As ψ_1 commutes with all ψ_i so do $g(\psi_1)$ and $h(\psi_1)$. Therefore, both $\operatorname{Ker} g(\psi_1)$ and $\operatorname{Ker} h(\psi_1)$ are invariant subspaces of all ψ_i:

$$0 = \psi_i(0) = (\psi_i \circ g(\psi_1))(v) = (g(\psi_1) \circ \psi_i)(v) \text{ for all } v \in \operatorname{Ker} g(\psi_1)$$

(analogously for $\operatorname{Ker} h(\psi_1)$). We can now continue inductively on these two invariant subspaces and this procedure terminates as V is finite dimensional. □

Note that the proof of lemma 4.40 is constructive and determining a decomposition of the vector space V computationally reduces to factorizing the characteristic polynomials. Once such a decomposition is determined one can perform a change of basis to block triangularize the morphisms $\psi_1, \ldots, \psi_n$. For our purpose computing this block triangularized matrices is not necessary. It suffices to keep track of the involved factors of the characteristic polynomial.

Algorithm 4.41: Testing Residual Nilpotence for free-abelian-by-abelian groups

Input: a finitely generated group G with normal subgroup $N \cong \mathbb{Z}^n$ and G/N abelian and corresponding representation $\varphi : G/N \to \mathrm{GL}(n, \mathbb{Z})$

Output: whether G is residually nilpotent or not

```
1 Compute a set of generators M_1, ..., M_k of Im φ.
2 Determine a decomposition U_1 ⊕ ... ⊕ U_l = ℚ^n as in 4.40.
3 Write f_{i,j} = p_{i,j}^{d_{i,j}} for the minimal polynomial of M_j|_{U_i} where p_{i,j} is
    irreducible.
4 for i = 1, ..., l do
5     Set α_{i,j} to a root of f_{i,j} for each j.
6     Set I_i = ⟨α_{i,j} − 1 | j = 1, ..., k⟩ in ℤ[α_{i,1}, ..., α_{i,k}].
7     if 1 ∈ I_i then
8         Return: G is not residually nilpotent.
9 Return: G is residually nilpotent.
```

Proof. By corollary 4.17 G is residually nilpotent if and only if $\langle M_i - 1 \mid i = 1, \ldots, k \rangle_{\mathbb{Z}[\operatorname{Im}\varphi]}$ contains a matrix with eigenvalue 1. Without loss of generality we can replace the M_j with a change of basis by $M_j' := S^{-1} M_j S$ where S denotes a transition matrix corresponding to the decomposition $U_1 \oplus \ldots \oplus U_l$. By construction the matrices $M_1', \ldots, M_k'$ are in block diagonal form. Looking for a matrix with eigenvalue 1 can therefore be done by considering their operation on each of the subspaces $U_1, \ldots, U_l$ separately. Fix one of the U_i. Now the $M_j'|_{U_i}$ commute so they can be simultaneously triangularized over $\mathbb{C}$. However, the eigenvalues of any $M_j'|_{U_i}$ are roots of the same monic irreducible

integral polynomial $p_{i,j}$. This implies that for any choice of eigenvalues $\alpha_{i,j}$ and $\alpha'_{i,j}$ of $M'_j|_{U_i}$ for $j = 1, \ldots, k$ we have

$$1 \in \langle \alpha_{i,j} - 1 \mid j = 1, \ldots, k \rangle_{\mathbb{Z}[\alpha_{i,1}, \ldots, \alpha_{i,k}]} \iff 1 \in \langle \alpha'_{i,j} - 1 \mid j = 1, \ldots, k \rangle_{\mathbb{Z}[\alpha'_{i,1}, \ldots, \alpha'_{i,k}]}$$

as $\mathbb{Z}[\alpha_{i,1}, \ldots, \alpha_{i,k}] = \mathbb{Z}[\alpha'_{i,1}, \ldots, \alpha'_{i,k}]$ and one can just apply any morphism of the Galois group of $\mathbb{Q}[\alpha_{i,1}, \ldots, \alpha_{i,k}]/\mathbb{Q}$ mapping the $\alpha_{i,j}$ to $\alpha'_{i,j}$. So it suffices to check the condition once for a choice of $\alpha_{i,j}$. If 1 is contained in the ideal I_i, then there is an element in the image of the augmentation ideal with eigenvalue 1 and G is not residually nilpotent by 4.17. Conversely, if 1 is not contained in the ideal for each block, then G has to be residually nilpotent by 4.17. □

4.5. Computing Residual Nilpotent Subgroups Of Finite Index

We finish off this chapter with an algorithm that computes a residually nilpotent normal subgroup of finite index for any polycyclic group. In [Seg83, ch. 1.C, thm. 4, p. 19 f] the author showed that all polycyclic groups are (residually nilpotent)-by-finite by showing that they contain a normal subgroup of finite index which is residually a finite p-group for arbitrary primes p. (Analogously to definition 4.2 a group is called a *residually finite p-group* if the intersection $\bigcap\{N \trianglelefteq G \mid G/N \text{ a finite } p\text{-group}\}$ is trivial.) The proof is actually constructive and leads to the algorithm 4.43 stated below. For its correctness we need the following lemma:

4.42. Lemma
Let G be a polycyclic group and U a normal subgroup of finite index. Choose a set of generators $g_1, \ldots, g_n$ of G. Then $\bigcap_{i=1}^{n} U^{g_i} \leq U$ is a normal subgroup of G of finite index. We call this intersection the *core* of U in G.

Proof. See [Seg83, ch. 1.A, lemma 2, p. 2] for a more general version of this statement. □

The core of a subgroup can be computed as an intersection by the means discussed in chapter 3.

Algorithm 4.43: Residually nilpotent subgroup

Input: a polycyclic group G and a prime number p
Output: a residually finite p-subgroup U normal in G of finite index

1 Compute a poly-C_∞ normal subgroup H of finite index in G.
2 Determine a polycyclic sequence $h_1, \ldots, h_n$ of H and set $H_i = \langle h_i, \ldots, h_n \rangle$.
3 Initialize U as H_n.
4 **for** $i = n-1, \ldots, 1$ **do**
5 Replace U by the core of U in H_i.
6 Set V to $\gamma_2(U)U^p$ where U^p is the subgroup generated by the p-th powers of $u \in U$.
7 Compute the order e of the automorphism given by the action of h_i on U/V.
8 Determine the core of $U\langle h_i^e \rangle$ in H_i and replace U by it.
9 Return the core of U in G.

We give a slightly adjusted version of Segal's proof as we have stated the algorithm in an iterative version.

Proof. First note that every polycyclic group is poly-C_∞-by-finite (see for instance [Seg83, ch. 1.A, prop. 2, p. 2]). The poly-C_∞ normal subgroup H of G can be computed by means discussed in [Eic00, 9.7]. By lemma 4.42 we only need to show that the algorithm constructs a residually finite p-subgroup of H. For this we show that U will be a residually finite p-group at the end of each iteration of the loop. Note it follows that U is a residually finite p-group at the beginning of each iteration: First U is initialized as an infinite cyclic group and for every succeeding iteration this follows recursively. As subgroups of residually finite p-groups are residually finite p-groups, replacing U by its core does not change this property. Hence we only need to show that $U\langle h_i^e \rangle$ is a residually finite p-group and the claim follows.

For this we define $V_j := \gamma_j(U)U^{p^{j-1}}$ for $j \geq 2$. Then the quotient U/V_j is a finite p-group. As U is residually a finite p-group it follows that $\bigcap_{j=2}^{\infty} V_j = 1$. Now $U/V = U/V_2$ is finite so $e < \infty$ is well-defined and we have

$$\mathrm{C}_{\langle h_i \rangle}(U/V) = \langle h_i^e \rangle.$$

By viewing the elements of $\langle h_i^e \rangle / \mathrm{C}_{\langle h_i^e \rangle}(U/V_j)$ as automorphisms on U/V_j we can apply a proposition by Segal [Seg83, ch. 1.B, prop. 13, p. 14] which tells us that for each j there is an m_j such that $h_i^{ep^{m_j}} \in \mathrm{C}_{\langle h_i^e \rangle}(U/V_j)$. This implies that

$$Q_j := U\langle h_i^{ep^{m_j}} \rangle / V_j = U/V_j \times \langle h_i^{ep^{m_j}} \rangle$$

is a direct product and therefore itself residually a finite p-group. As the Q_j are normal in $U\langle h_i^e \rangle / V_j$ of index p^{m_j} we get that for each j the quotient $U\langle h_i^e \rangle / V_j$

is residually a finite p-group by the third isomorphism theorem. Finally, since $\bigcap_{j=2}^{\infty} V_j = 1$ we conclude that $U\langle h_i^e \rangle$ is a residually finite p-group and the proof is finished. □

An immediate question now would be whether one can use this algorithm to check if G itself is residually nilpotent. However, in general the answer to this is no. The subgroup generated by the algorithm is typically a proper subgroup even in the case where G itself is poly-C_∞ and one does not need to compute any cores as the following example illustrates.

4.44. Example
Let $G = \mathcal{H}_1$ be the Heisenberg group on three generators $g_1,\ g_2,\ g_3$ defined as in example 2.40. Fix some prime number p. G is poly-C_∞ itself so we set $H = G$ and initialize U as $\langle g_3 \rangle$. We start with the first iteration. U is already normal in $\langle g_2,\ g_3 \rangle$ and we set $V = \gamma_2(U)U^p = \langle g_3^p \rangle$. Now g_2 act trivially on g_3 so we replace U by $\langle g_2,\ g_3 \rangle$ which is again already normal. In the second iteration we see that the action of g_1 on $U/\langle g_2^p,\ g_3^p \rangle$ has order p as $g_2^{g_1} = g_2 g_3$. So the algorithm outputs g_1^p, g_2, g_3 as the residually a finite-p normal subgroup of G.
However, G is nilpotent and therefore residually nilpotent itself. This shows that the algorithm does not necessarily output the largest possible subgroup which means that it cannot be used to check residual nilpotence itself even if G is poly-C_∞.

5. Summary And Open Problems

We want to give a brief summary to conclude this thesis and point out a few open problems and possible directions on how to proceed from there. In chapter 2 we have developed a new fast algorithm to compute induced sequences. As we have shown at the end of that chapter, it allows handling examples that are much larger in terms of Hirsch length and the number of subgroup generators. The algorithm given there was designed to work well for arbitrary polycyclic groups. The focus of future developments in this area probably lies in the development of different algorithms specialized on certain classes of polycyclic groups. This includes specialized preprocessing steps e.g. as discussed in 2.41.

As computing induced sequences is a core step in many algorithms on polycyclic groups, changing this step has a large impact on their performance. We have seen this in chapter 3 for the intersection of subgroups. This bolsters further investigations on how well these algorithms synergize with the new induced sequence method and the respective implementations optimized for the old version of the induced sequence algorithm should be revisited.

In chapter 4 we have discussed residual nilpotence of polycyclic groups and gave practical algorithms to determine whether a polycyclic [free-]abelian-by-(finite and nilpotent) or [free-]abelian-by-abelian group is residual nilpotent. It remains the case of [free-]abelian-by-(infinite nilpotent) groups. From a theoretical perspective, if the representation given by the action of the nilpotent quotient can be split into induced subrepresentations with either abelian or finite nilpotent image one can apply the methods discussed in this work (see lemma 4.20). However, algorithmically splitting representations over characteristic 0 is not a simple task which is why this remains still open. Additionally, as it was pointed out to us by Yves de Cornulier on mathoverflow[3], one can construct explicit examples of $\mathbb{R}$-irreducible infinite nilpotent integer represen-

[3] see `https://mathoverflow.net/questions/469209`

tations which are not abelian. As they are irreducible, they cannot be split and thus our algorithms do not apply to these examples and one needs to use a different approach to tackle them. However, he and David A. Craven also argued (loc. cit.) that there is no such example of an infinite nilpotent non-abelian integer representations which is $\mathbb{C}$-irreducible. So if we assume we can decompose such a representation over the complex numbers, we could try to use the algorithms described in this thesis to look for non-trivial fixvectors of an element in the augmentation ideal. This way one can show that a group is not residually nilpotent. However showing that such a group is residually nilpotent still requires additional work as the positive result of theorem 4.29 only applies to the case of integer matrices.

Another direction to take from here is to develop a general algorithm working for arbitrary polycyclic groups. Polycyclic groups are known to be (nilpotent-by-abelian)-by-finite (see for instance [Seg83, ch. 2.C thm. 4 p. 35]). So a possible approach to this might be applying the results of chapter 4 inductively on a suitable series. However, as we have seen in remark 4.7 whether a quotient is residual nilpotent or not does not necessarily tell us anything about the parent group. For this to work one has to develop some additional criteria like the one given in proposition 4.8. In [FR85] and [BNB21] the authors of these articles gave sufficient criteria for extensions of residual nilpotent groups to be residual nilpotent itself. However, the preconditions on those are still too restrictive to give results for arbitrary polycyclic groups.

At the end of chapter 4 we have seen how to compute a residual nilpotent subgroup of finite index in a polycyclic group. So another approach to verify that such a group G is residual nilpotent could be to determine residual nilpotent subgroups U and V of coprime index in G and give some criterion why the product UV itself remains residual nilpotent. Of course such an approach fails to show the contrary.

Appendix

Group Presentations

The appendix contains some polycyclic presentations which we used in examples and performance tests. We moved their formal definitions here for readability reasons.

The groups $\mathcal{G}_3$ and $\mathcal{V}_3$

The group $\mathcal{G}_3$ is given by the presentation:

$$\mathcal{G}_3 = \langle g_1, g_2, g_3 \mid g_2^{g_1^{\pm 1}} = g_2 g_3^3,\ g_3^{g_1^{\pm 1}} = g_3^{-1},\ g_3^{g_2^{\pm 1}} = g_3^{-1} \rangle.$$

This group is in particular interesting as it is a relatively small (Hirsch length 3) non-abelian, residually nilpotent but not nilpotent group in which one can construct simple examples in which the current implementation of the induced sequence algorithm in the GAP package Polycyclic produces wrong result. Such an example would be

$$\mathcal{V}_3 = \langle g_1^7 g_2^2 g_3^{-1},\ g_1^{11} g_2^{-2} g_3^{-10} \rangle.$$

The canonical induced sequence of $\mathcal{V}_3$ is $g_1 g_2^{26} g_3^8,\ g_2^{36} g_3^9,\ g_3^{18}$.

The groups $\mathcal{H}_n$ and $\mathcal{U}_n$

The Heisenberg group $\mathcal{H}_n$ for $n \in \mathbb{N}_{>0}$ was already defined in example 2.40 as

$$\begin{aligned} \langle g_1, \ldots, g_{2n+1} \mid & g_{n+i}^{g_i} = g_{n+i} g_{2n+1},\ g_{n+i}^{g_i^{-1}} = g_{n+i} g_{2n+1}^{-1} \text{ for } 1 \le i \le n, \\ & g_j^{g_i^{\pm 1}} = g_j \text{ else} \rangle. \end{aligned}$$

It is nilpotent and of Hirsch length $2n+1$. We used this group for different performance tests. In section 2.7 we also used its subgroup

$$\mathcal{U}_n = \langle g_1^2 \ldots,\ g_n^2, g_{n+1}^3, \ldots, g_{2n}^3,\ g_{2n+1}^6 \rangle$$

for this purpose.

The group $\mathcal{G}_9$

We define the group $\mathcal{G}_9$ given by the presentation

$$\begin{aligned}
\langle x_1, \ldots, x_9 \mid & x_1^2 = 1,\ x_2^4 = 1,\ x_3^3 = 1,\ x_4^5 = 1, && x_2^{x_1} = x_2^{x_1^{-1}} = x_2^3, \\
& x_5^{x_1} = x_5^{x_1^{-1}} = x_5^3 x_7^4 x_8^{-4} x_9^5, && x_6^{x_1} = x_5^{x_1^{-1}} = x_5^{108} x_6 x_7^{216} x_8^{-216} x_9^{270}, \\
& x_7^{x_1} = x_5^{x_1^{-1}} = x_5^{-60} x_7^{-119} x_8^{120} x_9^{-150}, && x_8^{x_1} = x_5^{x_1^{-1}} = x_5^2 x_7^4 x_8^{-3} x_9^5, \\
& x_9^{x_1} = x_5^{x_1^{-1}} = x_5^{48} x_7^{96} x_8^{-96} x_9^{121}, && \\
& x_5^{x_2} = x_5 x_7^{-4} x_9^{-5}, && x_6^{x_2} = x_5^{30} x_6 x_7^{-96} x_8^{-60} x_9^{-120}, \\
& x_7^{x_2} = x_5^{-18} x_7^{49} x_8^{36} x_9^{60}, && x_8^{x_2} = x_7^{-4} x_8 x_9^{-5}, \\
& x_9^{x_2} = x_5^{14} x_7^{-40} x_8^{-28} x_9^{49}, && x_5^{x_2^{-1}} = x_5^3 x_7^4 x_8^{-4} x_9^5, \\
& x_6^{x_2^{-1}} = x_5^{78} x_6 x_7^{216} x_8^{-156} x_9^{270}, && x_7^{x_2^{-1}} = x_5^{-42} x_7^{-119} x_8^{84} x_9^{-150}, \\
& x_8^{x_2^{-1}} = x_5^2 x_7^4 x_8^{-3} x_9^5, && x_9^{x_2^{-1}} = x_5^{34} x_7^{96} x_8^{-68} x_9^{121}, \\
& x_5^{x_3} = x_5^{-1} x_6^2 x_7^{-6} x_8^2 x_9^{-12}, && x_6^{x_3} = x_5^{-39} x_7^{-53} x_8^{87} x_9^{-66}, \\
& x_7^{x_3} = x_5^{25} x_7^{36} x_8^{-55} x_9^{45}, && x_8^{x_3} = x_5^{-1} x_6 x_7^{-3} x_8^2 x_9^{-6}, \\
& x_9^{x_3} = x_5^{-20} x_7^{-28} x_8^{44} x_9^{-35}, && x_5^{x_3^{-1}} = x_5^1 1 x_7^{14} x_8^{-22} x_9^{18}, \\
& x_6^{x_3^{-1}} = x_5^{-53} x_6^9 x_7^{-87} x_8^{107} x_9^{-129}, && x_7^{x_3^{-1}} = x_5^{30} x_6^{-5} x_7^{51} x_8^{-60} x_9^{75}, \\
& x_8^{x_3^{-1}} = x_5^5 x_7^7 x_8^{-10} x_9^9, && x_9^{x_3^{-1}} = x_5^{-24} x_6^4 x_7^{-40} x_8^{48} x_9^{-59}, \\
& x_j^{x_i^{\pm 1}} = x_j \text{ else} \rangle. &&
\end{aligned}$$

This is a (free-abelian)-by-(finite nilpotent) group which is residually nilpotent as shown in example 4.36.

The group $\mathcal{G}_6$

The group $\mathcal{G}_6$ is one of the example groups provided by the GAP package Polycyclic. It is returned by `ExamplesOfSomePcpGroups(2)`. This group is the

split extension of $\mathbb{Z}^4$ by $\mathbb{Z}^2$ where the action is given by

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 1 & 1 & 0 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \\ 1 & 1 & -1 & 0 \end{pmatrix}.$$

It is therefore abelian-by-abelian of Hirsch length 6. It is not residually nilpotent as can be verified by the techniques discussed in chapter 4.

The group $\mathcal{G}_{24}$

The group $\mathcal{G}_{24}$ is also one of the example groups provided by the GAP package Polycyclic. It is returned by `ExamplesOfSomePcpGroups(14)` which yields a polycyclic presentation on 24 generators and has Hirsch length 13. It is a nilpotent group of class 10 with torsion subgroup of order 2 250 000. We refrain from printing the presentation as its relations alone would fill more than two pages.

References

[AF11] Matthias Aschenbrenner and Stefan Friedl, *Residual properties of graph manifold groups*, Topology and its Applications **158** (2011), no. 10, 1179–1191.

[Aus67] Louis Auslander, *On a problem of Philip Hall*, Ann. Math. (2) **86** (1967), 112–116.

[Bau68] G. Baumslag, *On the residual nilpotence of certain one-relator groups*, Commun. Pure Appl. Math. **21** (1968), 491–506.

[BCR91] Gilbert Baumslag, Frank B. Cannonito, Derek J. S. Robinson, and Dan Segal, *The algorithmic theory of polycyclic-by-finite groups*, J. Algebra **142** (1991), no. 1, 118–149.

[Bea97] Robert Beals, *Towards polynomial time algorithms for matrix groups*, Groups and computation II. Workshop on groups and computation, June 7–10, 1995, New Brunswick, NJ, USA, Providence, RI: American Mathematical Society, 1997, pp. 31–54.

[BNB21] V. G. Bardakov, M. V. Neshchadim, and O. V. Bryukhanov, *On residually nilpotence of group extensions*, `https://doi.org/10.48550/arXiv.2106.11678` 2021.

[Bra71] Gordon H. Bradley, *Algorithms for Hermite and Smith normal matrices and linear diophantine equations*, Math. Comput. **25** (1971), 897–907.

[CMZ17] Anthony E Clement, Stephen Majewicz, and Marcos Zyman, *The theory of nilpotent groups*, 1st ed., Springer International Publishing AG, Cham, 2017.

[Coh93] Henri Cohen, *A course in computational algebraic number theory*, 1st ed., Graduate Texts in Mathematics 138, Springer Berlin Heidelberg, Berlin, Heidelberg, 1993.

[EDIM] Frank Lübeck, *EDIM GAP-package for Elementary Divisors of Integer Matrices, Version 1.3.8*, `https://github.com/frankluebeck/EDIM`, 2024.

[Eic00] Bettina Eick, *Algorithms for polycyclic groups*, Habilitationsschrift, Gesamthochschule Kassel, `http://www.icm.tu-bs.de/~beick/publ/habil.pdf`, 2000.

[Eic21] ———, *Computing the order and the index of a subgroup in a polycyclic group*, `https://doi.org/10.48550/arXiv.2102.04023`, 2021.

[FR85] Michael Falk and Richard Randell, *The lower central series of a fiber-type arrangement*, Invent. Math. **82** (1985), 77–88.

[Fru77] M. A. Frumkin, *Polynomial time algorithms in the theory of linear diophantine equations*, Fundamentals of Computation Theory (Berlin, Heidelberg) (Marek Karpiński, ed.), Springer Berlin Heidelberg, 1977, pp. 386–392.

[GAP] The GAP Group, *GAP – Groups, Algorithms, and Programming, Version 4.11.1*, `https://www.gap-system.org`, 2022.

[Hal69] Philip Hall, *Nilpotent groups : notes of lectures given at the Canadian mathematical congress, summer seminar, University of Alberta, 12 - 30 August, 1957*, Queen Mary College mathematics notes, Queen Mary College, Math. Dep., 1969.

[HEO05] Derek F. Holt, Bettina Eick, and Eamonn A. O'Brien, *Handbook of computational group theory*, Discrete mathematics and its applications, Chapman and Hall/CRC, Boca Raton, 2005.

[Hir38] K. A. Hirsch, *On infinite soluble groups. I*, Proc. Lond. Math. Soc. (2) **44** (1938), 53–60.

[Hir54] ———, *On infinite soluble groups. V*, J. Lond. Math. Soc. **29** (1954), 250–251.

[HJ10] Roger A. Horn and Charles R. Johnson, *Matrix analysis*, 1. paperback ed., 23. print. ed., Cambridge Univ. Press, Cambridge, 2010.

[HM98] George Havas, Bohdan S. Majewski, and Keith R. Matthews, *Extended GCD and Hermite normal form algorithms via lattice basis reduction*, Exp. Math. **7** (1998), no. 2, 125–136.

[Itô53] Noboru Itô, *Note on S-groups*, Proc. Japan Acad. **29** (1953), 149–150.

[KB79] Ravindran Kannan and Achim Bachem, *Polynomial algorithms for computing the Smith and Hermite normal forms of an integer matrix*, SIAM J. Comput. **8** (1979), 499–507.

[Kow19] Hans-Joachim Kowalsky, *Lineare Algebra*, Goschens Lehrbucherei/-Gruppe I: Reine und angewandte Mathematik, De Gruyter, 2019.

[Kru12] Krumke, Sven Oliver and Noltemeier, Hartmut, *Graphentheoretische Konzepte und Algorithmen*, 3. aufl. 2012, Leitfäden der Informatik, Vieweg+Teubner Verlag, Wiesbaden, 2012.

[KT06] Jon Kleinberg and Éva Tardos, *Algorithm design*, Pearson internat. ed., Pearson, Boston, 2006.

[LGS98] C. R. Leedham-Green and Leonard H. Soicher, *Symbolic collection using deep thought*, LMS journal of computation and mathematics **1** (1998), 9–24.

[Luk92] Eugene M. Luks, *Computing in solvable matrix groups*, 33rd annual symposium on Foundations of computer science (FOCS). Proceedings, Pittsburgh, PA, USA, October 24–27, 1992, Washington, DC: IEEE Computer Society Press, 1992, pp. 111–120.

[MAG] Wieb Bosma, John Cannon, and Catherine Playoust, *The Magma algebra system. I. The user language*, J. Symbolic Comput. **24** (1997), no. 3-4, 235–265, Computational algebra and number theory (London, 1993).

[Mag35] W. Magnus, *Beziehungen zwischen Gruppen und Idealen in einem speziellen Ring.*, Math. Ann. **111** (1935), 259–280.

[Mal40] A. I. Mal'cev, *On isomorphic matrix representations of infinite groups*, Rec. Math. Moscou, n. Ser. **8** (1940), 405–422 (original in Russian).

[Mal51] ———, *Über einige Klassen unendlicher auflösbarer Gruppen*, Mat. Sb., Nov. Ser. **28** (1951), 567–588 (original in Russian).

[McC96] James McCarron, *Residually nilpotent one-relator groups with non-trivial centre*, Proc. Am. Math. Soc. **124** (1996), no. 1, 1–5.

[Ost99] Gretchen Ostheimer, *Practical algorithms for polycyclic matrix groups*, Journal of Symbolic Computation **28** (1999), no. 3, 361–379.

[POLY] B. Eick, W. Nickel, and M. Horn, *Polycyclic, computation with polycyclic groups, Version 2.16*, `https://gap-packages.github.io/polycyclic/`, Jul 2020, Refereed GAP package.

[Rob82] Derek John Scott Robinson, *A course in the theory of groups*, Graduate texts in mathematics 80, Springer, New York, 1982.

[Sch84] Manfred R. Schroeder, *Number theory in science and communication : with applications in cryptography, physics, biology, digital information and computing*, Springer series in information sciences 7, Springer, Berlin, 1984.

[Seg83] Daniel Segal, *Polycyclic groups*, Cambridge tracts in mathematics 82, Cambridge Univ. Pr., Cambridge, 1983.

[Seg90] Dan Segal, *Decidable properties of polycyclic groups*, Proc. Lond. Math. Soc. (3) **61** (1990), no. 3, 497–528.

[Ser77] Jean-Pierre Serre, *Linear representations of finite groups*, 1st ed., Graduate Texts in Mathematics 42, Springer New York, New York, NY, 1977.

[Sim94] Charles C. Sims, *Computation with finitely presented groups*, vol. 48, Cambridge: Cambridge University Press, 1994.

[Swa67] R. G. Swan, *Representations of polycyclic groups*, Proc. Am. Math. Soc. **18** (1967), 573–574.

[Weh09] Bertram A. F. Wehrfritz, *Group and ring theoretic properties of polycyclic groups.*, Algebr. Appl., vol. 10, Dordrecht: Springer, 2009.

Lebenslauf

Ausbildung

seit 04/2020	**Promotionsstudium Mathematik** RPTU Kaiserslautern-Landau Betreuer: Prof. Dr. Max Horn
04/2017 – 10/2019	**Masterstudium Mathematik (M.Sc.)** TU Kaiserslautern
04/2014 – 09/2017	**Bachelorstudium Mathematik (B. Sc.)** TU Kaiserslautern

Berufserfahrung

seit 04/2020	**Wissenschaftlicher Mitarbeiter** Fachbereich Mathematik RPTU Kaiserslautern-Landau
10/2016 – 07/2019 und 04/2015 – 02/2016	**Wissenschaftliche Hilfskraft** Fachbereich Mathematik TU Kaiserslautern

Curriculum Vitae

Education

since 04/2020	**PhD Student in Mathematics** RPTU Kaiserslautern-Landau Supervisor: Prof. Dr. Max Horn
04/2017 – 10/2019	**Master of Science in Mathematics** TU Kaiserslautern
04/2014 – 09/2017	**Bachelor of Science in Mathematics** TU Kaiserslautern

Employment

since 04/2020	**Research Assistant** Department of Mathematics RPTU Kaiserslautern-Landau
10/2016 – 07/2019 and 04/2015 – 02/2016	**Student & Teaching Assistant** Department of Mathematics TU Kaiserslautern