Computing the order and the index of a subgroup in a polycyclic group

Bettina Eick

February 9, 2021

Abstract

This contains a new version of the so-called ‘non-commutative Gauss’ algorithm for polycyclic groups. Its results allow to read off the order and the index of a subgroup in an (possibly infinite) polycyclic group.

1 Introduction

Practical algorithms to compute with finite polycyclic groups have been described by Laue, Neubüser & Schoenwaelder in [3]. A main basis for most of these algorithms is the so-called ‘non-commutative Gauss’ algorithm. Given a finite polycyclic group $G$ and a finite set of generators for a subgroup $U$, this computes a so-called induced polycyclic generating sequence for $U$. In turn, this allows to read off $|U|$ and $[G:U]$ and forms a basis for many other algorithms.

Eick [2] introduced various practical algorithms to compute with possibly infinite polycyclic groups. These also included an extended version of the ‘non-commutative Gauss’ algorithm. The proof of this extended version has gaps. It is the main aim here to cover this open problem and to introduce a practical and reliable version of the ‘non-commutative Gauss’ algorithm for possibly infinite polycyclic groups.

A GAP implementation of the code described here is available online, see [1].

2 Preliminaries

We first introduce the setting of the algorithm. We assume that the group $G$ is given by a consistent polycyclic presentation; that is, it has generators $g_1,\ldots,g_n$ and non-negative integers $r_1,\ldots,r_n$ and relations of the form

\[
\begin{align*}
g_i g_j &= g_j^{a_{i,j}^1} \cdots g_n^{a_{i,j}^n} \text{ for } 1 \leq j < i \leq n, \\
g_i g_j^{-1} &= g_j^{-1} g_j^{b_{i,j}^1} \cdots g_n^{b_{i,j}^n} \text{ for } 1 \leq j < i \leq n \text{ with } r_i = 0, \\
g_i^{r_i} &= g_i^{c_i^1} \cdots g_n^{c_i^n} \text{ for } 1 \leq i \leq n \text{ with } r_i > 0,
\end{align*}
\]

where $a_{i,j,k}, b_{i,j,k}, c_{i,k}$ are integers that are contained in $\{0,\ldots,r_k - 1\}$ if $r_k > 0$. Additionally, it is required that for each element $g$ of $G$ there exists a unique $(e_1,\ldots,e_n) \in \mathbb{Z}^n$
with \( 0 \leq e_k < r_k \) if \( r_k > 0 \) so that
\[
g = g_1^{e_1} \cdots g_n^{e_n}.
\]

We call \( g_1^{e_1} \cdots g_n^{e_n} \) the normal form of the element \( g \). It can be computed readily for any arbitrary word in the generators by iteratively applying the relations of the group.

Let \( G_i = \langle g_i, \ldots, g_n \rangle \) for \( 1 \leq i \leq n \). The relations of \( G \) imply that \( G_{i+1} \leq G_i \) for \( 1 \leq i \leq n-1 \) and \( G_i/G_{i+1} \) is cyclic. The consistency of the presentation implies that \( [G_i : G_{i+1}] = r_i \) if \( r_i > 0 \) and \( [G_i : G_{i+1}] = \infty \) if \( r_i = 0 \).

We introduce some further notation for elements in a group \( G \) given by a consistent polycyclic presentation. Let \( g = g_1^{e_1} \cdots g_n^{e_n} \) be a normal form and assume that \( g \neq 1 \). Then

- the depth of \( g \) is \( d \) if \( e_1 = \ldots = e_{d-1} = 0 \) and \( e_d \neq 0 \). Write \( d(g) \).
- the leading exponent of \( g \) is \( e_d \). Write \( l(g) \).
- the relative order of \( g \) is \( |gG_{d+1}| \). Write \( r(g) \).

If \( g = 1 \), then we say that \( g \) has depth \( n + 1 \) and leading exponent or relative order of \( g \) do not exist. If \( g \neq 1 \), then \( r(g) = \infty \) if \( r_d = 0 \) and \( r(g) \mid r_d \) otherwise.

3 Subgroups, Igs and Cgs

Let \( G \) be given by a consistent polycyclic presentation. A generating set \( u_1, \ldots, u_m \) of a subgroup \( U \) is an igs if the series \( U_i = \langle u_1, \ldots, u_m \rangle \) with \( 1 \leq i \leq m \) coincides with the series \( G_i \cap U \) for \( 1 \leq i \leq n \) where duplicates have been removed. The following has been proved in [2].

1 Lemma: Let \( u_1, \ldots, u_m \) be a generating set for \( U \). Then \( u_1, \ldots, u_m \) is an igs for \( U \) if and only if

- \( u_i^{u_j} \in U_{j+1} \) for \( 1 \leq j < i \leq m \),
- \( u_i^{r(u_i)} \in U_{i+1} \) for \( 1 \leq i \leq m \) with \( r(u_i) > 0 \),
- \( d(u_1) < d(u_2) < \ldots < d(u_m) \).

Proof: We include a proof for completeness.

(1) First assume that \( u_1, \ldots, u_m \) is an igs. Choose \( j \) maximal with \( U_i = G_j \cap U \). Then \( U_{i+1} = G_{j+1} \cap U \) and thus \( U_{i+1} \leq U_i \) with \( U_i/U_{i+1} \) of order \( r(u_i) \). Thus all three items follow.

(2) Now assume that the three items are satisfied. By item (a) it follows that \( U_{i+1} \leq U_i \) and by construction and item (b) the quotient \( U_i/U_{i+1} \) is cyclic of order \( r(u_i) \) if \( r(u_i) > 0 \) and cyclic of order \( \infty \) if \( r(u_i) = 0 \). Induction now yields the desired result.

4 Computing an igs

We assume that generators \( h_1, \ldots, h_l \) for a subgroup \( U \) of \( G \) are given. Our aim is to determine an igs for \( U \).
4.1 Normalisations of elements

Let \( g \in G, \ g \neq 1 \), with depth \( d \), leading exponent \( a \) and relative order \( r_d \). If \( r_d = 0 \), then let \( e = \text{sign}(a) \) and call \( g^e \) the normalisation of \( g \). If \( r_d > 0 \), then write \( a = xy \) with \( x = \gcd(a, m) \) and \( y = a/x \). Note that \( z = y^{-1} \mod m \) exists and call \( g^z \) the normalisation of \( g \).

2 Remark: Let \( g \in G, \ g \neq 1 \), with depth \( d \) and normalisation \( h \).

(a) \( d(h) = d(g) \)
(b) \( l(h) \mid l(g) \).
(c) \( \langle g, G_{d+1} \rangle = \langle h, G_{d+1} \rangle \).

Remark 2 indicates why the normalisation of an element is of interest: in the cyclic group \( G_d/G_{d+1} \) it yields the unique generator of \( \langle gG_{d+1} \rangle \) with smallest leading exponent.

4.2 Partial Igs

A partial igs is a list of length \( n \) (the number of generators of the parent group \( G \)) whose \( i \)-th entry is either empty or a normalised element \( g \) in \( G \) of depth \( i \).

The following function takes a partial igs \( I \) and an arbitrary element \( g \in G \) and determines a new partial igs \( J \) so that \( \langle J \rangle = \langle I, g \rangle \).

AddGenToPIgs\((I, g)\):
(1) Initialise \( L \) as the list with a single entry \( g \).
(2) While \( L \) is not empty do:
   (a) Take an element \( h \) from \( L \) and eliminate it in \( L \).
   (b) Let \( d = d(h) \). If \( d > n \) then go back to (2).
   (c) If \( I[d] \) is empty then:
      (i) Insert the normalisation of \( h \) at position \( d \) in \( I \).
      (ii) If \( r_d > 0 \) then add \( h \cdot I[d]^{-l(h)/l(I[d])} \) to \( L \).
   (d) If \( I[d] \) is not empty then:
      (i) Let \( k = I[d] \) and \( b = l(k) \) and \( a = l(h) \).
      (ii) Let \( e = \gcd(a, b) = ua + vb \) and \( w = h^uk^v \).
      (iii) Insert the normalisation of \( w \) at position \( d \) in \( I \).
      (iv) Add \( h \cdot I[d]^{-l(h)/l(I[d])} \) to \( L \).
      (v) Add \( k \cdot I[d]^{-l(k)/l(I[d])} \) to \( L \).

First, note that in Steps (2d)(iv) and (2d)(v), the quotients \( l(h)/l(I[d]) \) and \( l(k)/l(I[d]) \) are integers, since \( l(I[d]) \mid l(w) = e = \gcd(a, b) \) and \( a = l(h) \) and \( b = l(k) \). Hence these Steps yield elements of \( G \) that are added to \( L \).

Second, in the Steps (2c)(ii), (2d)(iv) and (2d)(v) there are elements of \( G \) added to \( L \). All of these elements have depth greater than \( d \). This implies that the algorithm terminates eventually.
3 Lemma: Let $J = \text{AddGenToPIgs}(I, g)$. Then $J$ is a partial igs satisfying $\langle J \rangle = \langle I, g \rangle$.

Proof: $J$ is a partial igs, since we only add normalised elements at the places associated with their depth. It remains to prove $\langle J \rangle = \langle I, g \rangle$.

$\subseteq$: Each element that is inserted into $I$ during the algorithm is a product of elements of $I \cup \{g\}$. Hence $J \subseteq \langle I, g \rangle$ and this part follows.

$\supseteq$: We show that $\langle L, I \rangle$ does not change in the course of the algorithm. Since $I\cup\{g\} = I\cup L$ to begin with and $J = I \cup L$ at the end, this yields the desired result. We consider the changes made to $I$ and $L$ in the course of the algorithm. In Step (2a) we take an element $h$ from $L$. There are several cases:

(Case 1): $I[d]$ is empty and $r_d = 0$. Then we add $h$ or $h^{-1}$ to $I$ and the result follows.

(Case 2): $I[d]$ is empty and $r_d > 0$. Then we add the normalization $h^z$ to $I$ and $h \cdot (h^z)^{-q}$ to $L$ for some $q$. Hence $h = h \cdot (h^z)^{-q} \cdot (h^z)^q$ can be obtained from $L$ and $I$ and the result follows.

(Case 3): $I[d] = k$. Then we add the normalization of $(h^a k^v)$ to $I$ and suitable quotients of $h$ and $k$ to $L$. As in Case 2, the quotients yield that $h$ and $k$ can be recovered from $L$ and $I[d]$. Hence the result follows in this case also. •

We note two obvious improvements of the algorithm.

(1) If there exists $l \in \{1, \ldots, n\}$ so that $l(I[d]) = 1$ for $l \leq d \leq n$, then we can improve the break in Step (2b) to: 'If $d \geq l$ then go back to (2)'. We can also replace the elements in $I$ so that $I[d] = g_d$ for $d \geq l$.

(2) In Step (2d) we insert the normalization of $w$ only if its leading exponent is not equal to $b$. Further, if the leading exponent of the normalization of $w$ equals either $a$ or $b$, then only one left quotient needs to be added to $L$.

4.3 Computing an igs

The following algorithm takes a list $L$ of elements of $G$ and determines an igs for the subgroup they generate. The algorithm is based on Lemma 4.

**IgsByGenerators**$(L)$:

(1) Initialise $I$ as a list of length $n$ with empty entries.

(2) While $L$ is not empty do:

(a) Take an element $g$ from $L$ and eliminate it in $L$.

(b) Run $\text{AddGenToPIgs}(I, g)$.

(c) Let $N$ denote the list of changes to $I$ in (2b).

(d) For $g$ in $N$ do:

(i) If $r(g)$ is finite, then add $g^{r(g)}$ to $L$.

(ii) For $h$ in $I$ with $h \neq g$ add $[g, h]$ to $L$.

(3) Return $I$. 

4
The algorithm terminates, since the depths of the elements in \( L \) increases in each step. The algorithm determines an igs for \( \langle L \rangle \), since it returns a list that generates \( \langle L \rangle \) and satisfies the conditions of Lemma 1.

5 Computing the order and the index

Suppose that a subgroup \( U \) of \( G \) is given by a set of generators. Then \( U \) and \( [G : U] \) can both be read off from an igs of \( U \).

4 Lemma: Let \( u_1, \ldots, u_m \) be an igs for \( U \), let \( D = \{ d(u_i) \mid 1 \leq i \leq m \} \) and let \( \overline{D} = \{1, \ldots, n\} \setminus D \).
(a) \( |U| = r(u_1) \cdots r(u_m) \).
(b) \( [G : U] = l(u_1) \cdots l(u_m) \cdot \prod_{d \in \overline{D}} r_d \).

6 Testing equality of subgroups

Suppose that two subgroups \( U \) and \( V \) of \( G \) are given. We would like to have an effective test for \( U = V \). We say that an igs \( u_1, \ldots, u_m \) is canonical if the normal forms
\[
u_i = g_1^{e_{i1}} \cdots g_n^{e_{in}}
\]
satisfy that if \( d(u_k) = d \) then \( e_{id} \in \{0, \ldots, l(u_k) - 1\} \) for \( 1 \leq i \leq m \). It is not difficult to determine a canonical igs from an arbitrary one by replacing \( u_i \) by \( u_i \cdot u_k^{-q} \) for all \( i \) and \( k \) where \( e_{id} = l(u_k)q + r \) with \( 0 \leq r < l(u_k) \) is determined by division with remainder.

5 Lemma: Two subgroups \( U \) and \( V \) are equal if and only if their canonical igs coincide.

References

[1] B. Eick. Implementation of code. www.iaa.tu-bs.de/beick/soft/igs.gi (2021)

[2] B. Eick. Algorithms for polycyclic groups. Habilitationsschrift, Universität Kassel, 2001.

[3] R. Laue, J. Neubüser, and U. Schoenwaelder. Algorithms for finite soluble groups and the SOGOS system. In Computational Group Theory, pages 105 – 135, London, New York, 1984. (Durham, 1982), Academic Press.