Homology, Homotopy and Applications, vol.1, No.7, 1999, pp.163–168 ISSN 1512–0139

COMPUTING SECOND COHOMOLOGY OF FINITE GROUPS WITH TRIVIAL COEFFICIENTS

GRAHAM ELLIS AND IRINA KHOLODNA

(*communicated by Ronald Brown*)

Abstract

The paper describes a simple procedure for computing the second cohomology $H^2(G, A)$ of a finitely presented finite group G with coefficients in a finitely presented finite abelian group A. An accompanying text-file contains a MAGMA implementation of the procedure.

1. Introduction

This paper describes the mathematics behind a simple procedure for computing the second cohomology $H^2(G, A)$ of a group G with coefficients in an abelian group A, where G acts trivially on A. An accompanying text-file contains computer code for adding the function

$$\mathtt{S},\ \mathtt{Cocycles},\ \mathtt{Extensions},\ \mathtt{InducedPhi} := \mathtt{SecondCohomology}(\mathtt{G},\mathtt{A},\mathtt{Phi},\mathtt{q},\mathtt{t})$$

to the computer-algebra language MAGMA V2.4 [**1**]. The input variables for this function are: a finitely presented finite group `G`; a finitely presented finite abelian group `A`; any homomorphism $\mathtt{Phi}\colon \mathtt{G} \to \mathtt{G}$ or homomorphism $\mathtt{Phi}\colon \mathtt{A} \to \mathtt{A}$; natural numbers `q` and `t`. In general one sets $\mathtt{q} := 1$, but when both G and A are p-groups the assignment $\mathtt{q} := p$ leads to better performance. The parameter `t` is a boolean toggle. One sets $\mathtt{t} := 1$ if information on cocycles is required, and $\mathtt{t} := 0$ in order to omit the computation of cocycles. The first output variable `S` is a sequence $\{_{m_1}A, \cdots, {}_{m_t}A, A_{k_1}, \cdots, A_{k_d}\}$ of abelian groups whose direct product H is isomorphic to $H^2(G, A)$ via isomorphism (7) below.

The second output variable is a sequence `Cocycles`. If $\mathtt{t} = 1$, the ith term of the sequence is a mapping $\alpha_i\colon \mathtt{G} \times \mathtt{G} \longrightarrow \mathtt{A}$, namely a cocycle representing the ith element in H. If $\mathtt{t} = 0$ the sequence is empty. The third output variable is a sequence `Extension` whose ith term is a homomorphism of finitely presented groups $\eta_i\colon E_i \twoheadrightarrow \mathtt{G}$, namely a central extension representing the ith element in H. The fourth output variable is a mapping `InducedPhi` on the set of integers $\{1, \cdots, n\}$ where n denotes the order of H; the homomorphism $H \to H$ induced by `Phi` maps the ith element of H to the i'th element where $i' = \mathtt{InducedPhi}(i)$. Note that if `Phi` happens to be an isomorphism then the extension group E_i is isomorphic to the extension group $E_{i'}$. The output variable `InducedPhi` is the only output variable to depend on `Phi`.

The function also returns two further output variables. The first of these is the sequence of invariants for the abelian group $H^2(G, A)$. The second is a sequence `V`. The elements of `V` are tuples $(i_1, \cdots, i_{t+d})$ of integers, and the (bijective) correspondence $H \to \mathtt{V}$, $({}_{m_1}A[i_1], \cdots, A_{k_d}[i_{t+d}]) \mapsto (i_1, \cdots, i_{t+d})$ indicates the ordering on H; here we have written $B[i]$ to denote the ith element of a structure B (and this makes sense since each member B in the sequence `S` is a group whose elements are ordered.)

A GAP [**4**] implementation of the procedure is available on request from the second author. Although originally implemented as a first step towards calculations of higher-dimensional

Received 18 March 1999, revised 17 June 1999; published on 30 June 1999.
1991 Mathematics Subject Classification: 20J06
Key words and phrases: second cohomology, group extensions, Magma software
© 1999, Graham Ellis and Irina Kholodna. Permission to copy for private use granted.

cohomology with non-trivial coefficients [**2**], we feel that the procedure is likely to be of interest in its own right (*cf.* [**3**]). It makes implicit use of the Universal Coefficient Theorem (see (3) below) to reduce the computation of second cohomology to that of second integral homology. Both MAGMA and GAP contain the `darstellungsgruppe` function for constructing covering groups from which second integral homology can be extracted. However, our implementation takes a slightly different approach to the construction of covering groups which involves the LLL algorithm [**8**] for finding bases in integer lattices. This approach 'minimizes the number of generators' and 'maximizes the functoriality' of the relevant covering group.

2. The method

Let G be a group which is freely presented as a quotient $G \cong F/R$ of a free group F by a normal subgroup R. Let A be an abelian group. The five-term exact sequence in group cohomology [**5**] yields an isomorphism

$$H^2(G,A) \cong \text{cokernel}\,\{\text{Hom}(\frac{F}{[R,F]},A) \to \text{Hom}(\frac{R}{[R,F]},A)\} \tag{1}$$

where $\text{Hom}(-,A)$ denotes the group of group homomorphisms. Since $R/(R\cap[F,F])$ is free abelian (it is a subgroup of the free abelian group $F/[F,F]$), the canonical quotient homomorphism $R/[R,F] \twoheadrightarrow R/(R\cap[F,F])$ has a non-canonical splitting $\sigma\colon R/(R\cap[F,F]) \to R/[R,F]$. We thus have an isomorphism

$$\frac{R}{[R,F]} \cong \frac{R\cap[F,F]}{[R,F]} \oplus \sigma(\frac{R}{R\cap[F,F]})\,. \tag{2}$$

We set $M(G) = R\cap[F,F]/[R,F]$ and note that $M(G)$ is isomorphic to the second integral homology of G. Isomorphisms (1) and (2) combine to give

$$H^2(G,A) \cong \text{Hom}(M(G),A) \oplus \text{coker}\,\{\text{Hom}(\frac{F}{[F,F]},A) \to \text{Hom}(\frac{R}{R\cap[F,F]},A)\}\,. \tag{3}$$

Furthermore, letting $e = \exp(A)$ denote the exponent of A, we have

$$\text{Hom}(M(G),A) \cong \text{Hom}(M(G)/M(G)^e,A) \tag{4}$$

where $M(G)/M(G)^e = M(G)\otimes_{\mathbf{Z}} C_e$. (Here C_e denotes the cyclic group of order e. If A has infinite exponent then we take $e = 0$ and interpret C_0 as the infinite cyclic group.) If G is finitely generated, then G^{ab} and $M(G)/M(G)^e$ are both direct products of finitely many cyclic groups, say

$$G^{ab} \cong C_{k_1} \times \cdots \times C_{k_d}\,, \tag{5}$$

$$M(G)/M(G)^e \cong C_{m_1} \times \cdots \times C_{m_t}\,. \tag{6}$$

For each natural number n and abelian group A we define the quotient group $A_n = A/ < a^n \,:\, a\in A >$ and subgroup ${}_nA = \{a\in A : a^n = 1\}$. One readily checks that (3), (4), (5) and (6) combine to give an isomorphism

$$H^2(G,A) \quad\cong\quad {}_{m_1}A \times\cdots\times {}_{m_t}A \quad\times\quad A_{k_1}\times\cdots\times A_{k_d}\,. \tag{7}$$

Isomorphism (7) is the basis of our computer procedure.

The procedure reads in presentations for $G = F/R$ and A, and also the action of an endomorphism ϕ on the generators of either G or A. It uses the LLL algorithm to find a basis for $R/R\cap[F,F]$ expressed in terms of the generators of F. This basis and the presentation for G could be used to construct a presentation for a covering group D whose centre contains the Schur multiplier $M(G)$ and whose quotient $D/M(G)$ is isomorphic to G. However, instead of constructing D the procedure assumes that A is finite, uses the presentation of A to determine $e = \exp(A)$, and constructs a presentation for $D_e = D/M(G)^e$. The abelian group

$M(G)/M(G)^e$ is determined by applying to D_e one of MAGMA's procedures for solving the word problem in a finitely presented group. The basis for $R/R \cap [F,F]$ and the structure of $M(G)/M(G)^e$ are then used to calculate the required properties of $H^2(G,A)$ via isomorphism (7). In order to solve the word problem in D_e the procedure assumes that the group G is finite. This assumption ensures that $M(G)$ is finite, and hence that D_e is finite. If no information is known about G, other than its finiteness, then the word problem is solved using the Todd-Coxeter procedure; this option is specified by setting the parameter $\mathtt{q} := 1$. If G is a finite p-group, then so too are $M(G)$ and D_e. The p-quotient algorithm (which converts an arbitrary finite presentation of a finite p-group into a power-commutator presentation) is then used to solve the word problem in D_e; this option is specified by setting the parameter $\mathtt{q} := p$. More generally, if G is a finite soluble group then so too is D_e, and one could in principle use the soluble quotient algorithm to solve the word problem in D_e. This option is not included in the accompanying implementation of the procedure. The help files in the current version of Magma mention that a future release of the package will contain a function for converting an arbitrary finite presentation of a soluble group into a power-commutator presentation. When this release is available the 'soluble quotient option' can be included, under the setting $\mathtt{q} := 4$ say, by adding three lines of code to the subsidiary function `EnumeratedGroup` which is described below.

If an isomorphism $G \cong P$ is known between the finite group G and a permutation group P, then the extremely efficient methods of D.F. Holt [**6**][**7**] could be used to determine the Schur multiplier $M(G)$ and cohomology group $H^2(G,A)$. These methods are included as part of the standard MAGMA package (for A an elementary abelian p-group with possibly non-trivial G-action) and so are not included as an option in our procedure. (It should be noted that MAGMA's function for obtaining a faithful permutation representation of a finitely presented group is based on the Todd-Coxeter procedure.)

The procedure can be adapted to handle certain infinite groups G with finite coefficients A. The point here is that the right-hand side of isomorphism (7) is a finite abelian group if A is finite and G is finitely presented. The above method for determining the groups A_{k_i} is based on the LLL algorithm and does not require finiteness of G. However, the above method for determining the ${}_{m_i}A$ involves calculating the finite group $M(G)/M(G)^e$ by solving the word problem in D_e. If G is infinite then soo too is D_e. However, suppose that for some integer k depending on e the group G has the following property.

PROPERTY: The quotient G/G^{e^k} is a finite group, and for all $g \in G$ the identity $g^{e^i} = 1$ holds for some integer i only if the identity $g^{e^{k-1}} = 1$ holds.

(Here G^n denotes the normal subgroup of G generated by all elements g^n with $g \in G$.) This property implies that $M(G)/M(G)^e$ maps injectively into the quotient group $D_e/(D_e)^{e^k}$, and that this quotient group is finite. We could thus apply the Todd-Coxeter procedure (or possibly the p-quotient algorithm or soluble quotient algorithm) to a presentation of $D_e/(D_e)^{e^k}$ in order to determine $M(G)/M(G)^e$, and hence determine $H^2(G,A)$.

3. The implementation

The computer algebra language MAGMA is extremely readable. The code listed in the accompanying text-file can thus be viewed as a type of "pseudo code" giving fine details of our procedure. (It can, of course, also be loaded into MAGMA V2.4 and used!) The code consists of one main function and several subsidiary functions. The main function `SecondCohomology(G,A,Phi,q,t)` has inputs and outputs as described above. Suppose that $< \underline{x} \mid \underline{r} > = < x_1, \cdots, x_d \mid r_1, \cdots, r_m >$ is the presentation of the finite group G, so that $\underline{x}$ is a generating set for F and $\underline{r}$ is a set of elements in F that normally generate R. The function `RelatorMatrix(F,rels)`, with $\mathtt{rels} := \underline{r}$, uses the LLL algorithm to return integer matrices

B, T. The ith row of B, and the free group element

$$s_i := r_1^{T[i][1]} r_2^{T[i][2]} \cdots r_t^{T[i][m]},$$

both represent the ith element in a basis for $R/R\cap[F,F]$. This function is used to construct the set $\underline{s} := \{s_1, \cdots, s_d\}$. The function `EnumeratedGroup(A,q)` is applied to the finitely presented group A and returns an *enumerated* group $\overline{A}$ together with isomorphisms $\nu\colon A \to \overline{A}$, $\nu'\colon \overline{A} \to A$. If $q = 1$ the Todd-Coxeter procedure is used in the enumeration. If $q = p$ then the p-quotient algorithm is used. The function does not assume that A is abelian and can thus be applied to an arbitrary finite presentation. (The reader may wish to improve this function. As explained above, the soluble quotient algorithm could be included under the setting $q = 4$. MAGMA's implementations of the Todd-Coxeter procedure and p-quotient algorithm allow the user to set various parameters. The function uses the default settings, but for certain groups it may be useful to modify the function to allow other settings.) The main function computes $e := \exp(\overline{A})$ and constructs the presentations

$$\mathcal{P} \;=\; < \underline{x} \mid \underline{s} \cup \{[x,r] \;:\; x \in \underline{x}, r \in \underline{r}\} \cup \{r^e \;:\; r \in \underline{r}\} >,$$

$$\mathcal{P}' \;=\; < \underline{x} \mid \underline{s} \cup \{[x,y] \;:\; x,y \in \underline{x}\} >$$

where $[x,y]$ denotes a commutator. It is readily checked that $\mathcal{P}$ presents the group D_e described in the preceding section, and that $\mathcal{P}'$ presents G^{ab}. The function `EnumeratedGroup(`D_e`,q)` is used to construct an enumerated group $\overline{D}_e$. The subgroup of $\overline{D}_e$ generated by the images of the relators $\underline{r}$ is calculated; this subgroup is readily seen to be isomorphic to $M(G)/M(G)^e$. Working in $\overline{D}_e$, the main function determines the orders m_i of the generators of $M(G)/M(G)^e$. It also uses the presentation $\mathcal{P}'$ to produce an enumerated version $\overline{G}^{ab}$ of G^{ab}. The order k_i of the image in $\overline{G}^{ab}$ of each $s_i \in \underline{s}$ is computed. The groups A_{k_i}, ${}_{m_i}A$ in the right-hand side of isomorphism (7) are constructed as subgroups of the enumerated group $\overline{A}$ using the functions `NQuotient(`$\overline{A}$`,k)`, `NCoQuotient(`$\overline{A}$`,m)`. In order to construct a group extension η_v corresponding to a cohomology class $v \in H^2(G,A)$ one can use isomorphism (1) to represent v by a homomorphism $\nu\colon R/[R,F] \to A$. One can then form the group

$$E = \frac{(F/[R,F] \times A)}{< (r, \nu(r)^{-1}) \;:\; \text{for each generator } r \text{ of } R/[R,F] >}.$$

It is routine to check that v is represented by a central extension of the form $A \hookrightarrow E \twoheadrightarrow G$. The function `Extension` returns the homomorphism $E \twoheadrightarrow G$ of finitely presented groups. The function `Cocycle` uses `EnumeratedGroup(E,q)` to produce a surjection $\overline{E} \twoheadrightarrow G$ where $\overline{E}$ is an enumerated version of E. It then computes a corresponding cocycle $\alpha_v\colon G \times G \to A$ by choosing a set-theoretic section $\sigma_v\colon G \to \overline{E}$ and using the standard formula

$$\sigma_v(gh) = \sigma_v(g)\sigma_v(h)\alpha_v(g,h)\,.$$

The section σ_v is constructed as a composite map $G \to \overline{G} \to E$ where $\overline{G}$ is an enumerated version of G obtained from the function `EnumeratedGroup(G,q)`. The point of passing through $\overline{G}$ is to ensure that each element in the finitely presented group G first gets reduced to a canonical form. The endomorphism $\overline{\phi} : H^2(G,A) \to H^2(G,A)$ induced by `phi` is constructed in the function `InducedPhi`. The construction is based on a routine analysis of (7) and is divided into two cases. The case where `Phi` is an endomorphism on A is handled by the function `InducedPhiOnA`. The case where `Phi` is an endomorphism on G is handled by the function `InducedPhiOnG`. The second case involves finding integer solutions X to an equation $Y = XB$ where B is an integer matix and Y an integer vector. The function `SolveYeqXB(Y,B)` uses the LLL algorithm to produce the solution vector X.

4. An example

Having loaded our implementation into MAGMA V2.4, the commands

```
F:=FreeGroup(2); x:=F.1; y:=F.2; q:=2; t:=0;
G:=quo<F | x^2, y^256, (x*y)^2 >; A:=quo<F | x^2, y^4, (x,y) >;
_,_,Extensions,_,Invariants:=
            SecondCohomology(G,A,IdentityHomomorphism(A),q,t);
Invariants; [ Domain(Extensions[i]) : i in [1..#S]];
```

first list the abelian group invariants for $H^2(D_{256}, \mathbf{Z_2} \times \mathbf{Z_4})$ where D_{256} is the dihedral group of order 512, and then list presentations for the 64 Yoneda inequivalent extensions representing the cohomology classes of this cohomology group. A Sun Microsystems Ultra 10 workstation took about 0.5 seconds of CPU time to perform these commands. (The same commands, but with $\mathtt{q} := 1$, took about 2.0 seconds of CPU time.) Further commands, such as

```
Phi:=hom<A->A | [A.1,A.1*A.2]>;
Psi:=hom<G->G | [G.1*G.2,G.2^-1]>;
_,_,_,InducedPhi:=SecondCohomology(G,A,Phi,q,t); InducedPhi(7);
_,_,_,InducedPhi:=SecondCohomology(G,A,Psi,q,t); InducedPsi(7);
```

can help in the task of determining the distinct isomorphism classes represented by the groups in the sequence `E`. These particular `Phi` and `Psi` represent non-trivial automorphisms. The values of `InducedPhi(7)` and `InducedPsi(7)` are computed to be 15 and 8. Therefore `E[7]`, `E[8]` and `E[15]` are isomorphic groups.

If the Todd-Coxeter process is to be used (q=1) then the group G must be of a fairly modest order. For instance, with A as before, the command

```
_,_,Extensions:=SecondCohomology(G,A,IdentityHomomorphism(A),1,0);
```

took 15 secs of CPU time on the dihedral group

$$G := < x, y \mid x^2, y^{1000}, (xy)^2 >$$

of order 2000, and took 8 mins 14 secs of CPU time on the group

$$G := < x, y, z \mid x^{-1}y^3x^{-1}y, y^{-1}z^3y^{-1}z, x^2z^{-1}x^5z^{-1} >$$

of order 61 440. Most of the time is taken in the application of the Todd-Coxeter process to the group $D_e = D/M(G)^e$. Experimentation suggests the the presentation of D_e constructed in the procedure is often not the most 'efficient', and that Tietze transformations can be applied to obtain a 'more efficient' presentation. In order to include such transformations into the procedure one would need a MAGMA function

$$\mathtt{SD}, \mathtt{alpha}, \mathtt{beta} := \mathtt{Simplify}(\mathtt{D})$$

that called a finitely presented group D, applied Tietze transformations, and returned an isomorphic finitely presented group SD together with isomorphisms $\alpha: D \to SD$, $\beta: SD \to D$.

Acknowledgements.

The first author was based at the Max-Planck-Institute für Mathematik, Bonn during part of this work and would like to thank the institute for its generous hospitality. The second author is grateful to Forbairt for financial support.

References

1. W. Bosma, J. Cannon and C. Playoust, 'The MAGMA algebra system I: the user language', *J. Symbolic Comput.* **24** (1997), 235-265.

2. G. Ellis and I. Kholodna, 'Three-dimensional presentations for the groups of order at most 30', LMS J. Comp. Math., to appear.

3. D.L. Flannery and E. O'Brien, 'Computing 2-cocycles for central extensions and relative difference sets', *Communications in Algebra*, to appear.

4. The GAP Group, 'GAP - Groups, Algorithms and Programming, Version 4.1', School of Mathematical and Computational Sciences, University of St. Andrews, Scotland (1998).

5. P.J. Hilton and U.Stammbach, *A course in homological algebra*, Graduate Texts in Math. 4, Springer-Verlag (1970).

6. D.F. Holt, 'The calculation of the Schur multiplier of a permutation group', *Computational Group Theory, (Durham, 1982)*, Academic Press (1984), 307-318.

7. D.F. Holt, 'The mechanical computation of first and second cohomology groups', *J. of Symbolic Computation* 1 (1985), 351–361.

8. A.K. Lenstra, H.W. Lenstra and L. Lovasz, 'Factoring polynomials with rational coefficients', *Math. Annalen* 261 (1982), 515-534.

This article may be accessed via WWW at http://www.rmi.acnet.ge/hha/ or by anonymous ftp at ftp://ftp.rmi.acnet.ge/pub/hha/volumes/1999/n7/n7.(dvi,ps,dvi.gz,ps.gz)

Graham Ellis `Graham.Ellis@nuigalway.ie`

Max-Planck-Institut für Mathematik
Bonn

Irina Kholodna

Department of Mathematics
National University of Ireland
Galway