HAP

Homological Algebra Programming

A proposed GAP 4 package

by

Graham Ellis
Mathematics Department
NUI Galway

Contents

1 Introduction

The GAP 4 package HAP provides a library of functions for computations related to the cohomology of groups. Both finite and infinite groups are handled, with main emphasis on integer coefficients. The main manual for the library is available only in html format and can be accessed from the file pkg/Hap/www/index.html. A summary of the manual is provided below.

HAP can be used to make basic calculations in the cohomology of finite and infinite groups. For example, to calculate the integral homology $H_n(D_{201}, \mathbf{Z}) = \mathbf{Z_{402}}$ of the dihedral group of order 402 in dimension $n = 99$ we could perform the following commands.

```
gap> F:=FreeGroup(2);; x:=F.1;; y:=F.2;;
gap> G:=F/[x^2,y^201,(x*y)^2];; G:=Image(IsomorphismPermGroup(G));;
gap> GroupHomology(G,99);
[ 2, 3, 67 ]
```

The HAP command GroupHomology(G,n) returns the abelian group invariants of the n-dimensional homology of the group G with coefficients in the integers $\mathbf{Z}$ with trivial G-action.

The above example has two features that dramatically help the computations. Firstly, D_{201} is a relatively small group. Secondly, D_{201} has periodic homology with period 4 (meaning that $H_n(D_{201}, \mathbf{Z}) = H_{n+4}(D_{201}, \mathbf{Z})$ for $n > 0$) and so the homology groups themselves are small.

Typically, the homology of larger non-periodic groups can only be computed in low dimensions. The following commands show that:

the alternating group A_7 (of order 2520) has $H_{10}(A_7, \mathbf{Z}) = Z_6 \oplus (Z_3)^2$.

the special linear group $SL_3(\mathbf{Z_3})$ (of order 5616) has $H_8(SL_3(\mathbf{Z}_3), \mathbf{Z}) = \mathbf{Z}_6$.

the group $SL_3(\mathbf{Z}_5)$ (of order 372000) has $H_3(SL_3(\mathbf{Z}_5), \mathbf{Z}) = \mathbf{Z}_{24}$.

the abelian group $G = C_2 \times C_4 \times C_6 \times C_8 \times C_{10} \times C_{12}$ (of order 46080) has $H_6(G, \mathbf{Z}) = (\mathbf{Z2})^{280} \oplus (\mathbf{Z}_4)^{12} \oplus (\mathbf{Z}_{12})^3$
.

```
gap> GroupHomology(AlternatingGroup(7),10);
[ 2, 3, 3, 3 ]

gap> S:=Image(IsomorphismPermGroup(SL(3,3)));;
gap> GroupHomology(S,8);
[ 2, 3 ]

gap> S:=Image(IsomorphismPermGroup(SL(3,5)));;
gap> GroupHomology(S,3);
[ 8, 3 ]
```

```
gap>G:=AbelianGroup([2,4,6,8,10,12]);;
gap> GroupHomology(G,6);
[ 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
  2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
  2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
  2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
  2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
  2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
  2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
  2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
  2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
  2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
  2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
  2, 2, 2, 2, 2, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 12, 12, 12 ]
```

The command GroupHomology() returns the mod p homology when an optional third argument is set equal to a prime p. The following shows that the Sylow 2-subgroup P of the Mathieu simple group M_{24} has 6-dimensional mod 2 homology $H_6(P,\mathbf{Z}_2) = (\mathbf{Z}_2)^{1}43$. (The group P has order 1024 and the computation takes over two hours to complete.))

```
gap> GroupHomology(SylowSubgroup(MathieuGroup(24),2),6,2);
143
```

The homology of certain infinite groups can also be calculated. The following shows that the classical braid group B on eight strings (represented by a linear Coxeter diagram D with seven vertices) has 5-dimensional integral homology $H_5(B,\mathbf{Z}) = \mathbf{Z}_3$.

```
gap> D:=[  [1,[2,3]],  [2,[3,3]],  [3,[4,3]],  [4,[5,3]],  [5,[6,3]],  [6,[7,3]]  ];;

gap> GroupHomology(D,5);
[ 3 ]
```

The command GroupHomology(G,n) is a composite of several more basic HAP functions and attempts, in a fairly crude way, to make reasonable choices for a number of parameters in the calculation of group homology. For a particular group G you would almost certainly be better off using the more basic functions directly and making the choices yourself! Similar comments apply to functions for cohomology (ring) calculations. A summery of the basic HAP functions is given in the next chapter. For full details consult the html HAP manual.

2 List of HAP functions

The following is a list of the functions available in the HAP package. The html manual (avaiable in the directory pkg/Hap/www) gives full details.

1 ► `AddWords(v,w)`

Inputs two words v,w in a free ZG-module and returns their sum v+w. If the characteristic of Z is greater than 0 then the next function should be more efficient.

2 ► `AddWordsModP(v,w,p)`

Inputs two words v,w in a free ZG-module and the characteristic p of Z. It returns the sum v+w. If p=0 the previous function might be fractionally quicker.

3 ► `AlgebraicReduction(w)`
► `AlgebraicReduction(w,p)`

Inputs a word w in a free ZG-module and returns a reduced version of the word in which all pairs of mutually inverse letters have been cancelled. The reduction is performed in a free abelian group unless the characteristic p of Z is entered.

4 ► `CocycleCondition(R,n)`

Inputs a resolution R and an integer n¿0. It returns an integer matrix M with the following property. Suppose d=R.dimension(n). An integer vector $f = [f_1, \ldots, f_d]$ then represents a ZG-homomorphism $R_n \rightarrow Z_q$ which sends the i-th generator of R_n to the integer f_i in the trivial ZG-module Z_q (where possibly q=0). The homomorphism f is a cocycle if and only if $M^t f = 0$ mod q.

5 ► `Cohomology(X)`

Inputs either a cochain complex $X = C$ or a cochain map $X = (C \rightarrow D)$ over the integers Z. At present only characteristic 0 is implemented.

If $X = C$ then the torsion coefficients of $H_n(C)$ are retuned.

If $X = (C \rightarrow D)$ then the induced homomorphism $H_n(C) \rightarrow H_n(D)$ is returned as a homomorphism of finitely presented groups.

6 ► `CoxeterDiagramComponents(D)`

Inputs a Coxeter diagram D and returns a list $[D_1, \ldots, D_d]$ of the maximal connected subgraphs D_i.

7 ► `CoxeterDiagramDegree(D,v)`

Inputs a Coxeter diagram D and vertex v. It returns the degree of v (i.e. the number of edges incident with v).

8 ► `CoxeterDiagramFpArtinGroup(D)`

Inputs a Coxeter diagram D and returns the corresponding finitely presented Artin group.

9 ▶ `CoxeterDiagramFpCoxeterGroup(D)`

Inputs a Coxeter diagram D and returns the corresponding finitely presented Coxeter group.

10 ▶ `CoxeterDiagramIsSpherical(D)`

Inputs a Coxeter diagram D and returns "true" if the associated Coxeter groups is finite, and returns "false" otherwise.

11 ▶ `CoxeterDiagramMatrix(D)`

Inputs a Coxeter diagram D and returns a matrix representation of it. The matrix is given as a function DiagramMatrix(D)(i,j) where i,j can range over the vertices.

12 ▶ `CoxeterSubDiagram(D,V)`

Inputs a Coxeter diagram D and a subset V of its vertices. It returns the full sub-diagram of D with vertex set V.

13 ▶ `CoxeterDiagramVertices(D)`

Inputs a Coxeter diagram D and returns its set of vertices.

14 ▶ `Epicentre(G,N)`
▶ `Epicentre(G)`

Inputs a finite group G and normal subgroup N and returns a subgroup $Z^*(G,N)$ of the centre of N. The group $Z^*(G,N)$ is trivial if and only if there is a crossed module $d : E \to G$ with $N = Image(d)$ and with $Ker(d)$ equal to the subgroup of E consisting of those elements on which G acts trivially.

If no value for N is entered then it is assumed that $N = G$. In this case the group $Z^*(G,G)$ is trivial if and only if G is isomorphic to a quotient $G = E/Z(E)$ of some group E by the centre of E. (There are probably quicker ways to compute $Z^*(G,G)$.)

15 ▶ `EvaluateProperty(X,‘‘name’)’`

Inputs a record X (such as a ZG-resolution or chain map) and a string "name" (such as "characteristic" or "type"). It searches X.property for the pair ["name",value] and returns value. If X.property does not exist, or if ["name",value] does not exist, it returns fail.

16 ▶ `EvenSubgroup(G)`

Inputs a group G and returns a subgroup G+. The subgroup is that generated by all products xy where x and y range over the generating set for G stored by GAP. The subgroup is probably only meaningful when G is an Artin or Coxeter group.

17 ▶ `EquivariantChainMap(R,S,f)`

Inputs a ZG-resolution R, a ZG'-resolution S with G' finite, and a group homomorphism $f : G \to G'$. It outputs a record M with the following components:

M.source is the resolution R.

M.target is the resolution S.

M.mapping(w,n) is a function which gives the image in S_n, under a chain map induced by f, of a word w in R_n. (Here R_n and S_n are the n-th modules in the resolutions R and S.)

F.properties is a list of pairs such as ["type", "equivariantChainMap"].

The resolution S must have a contracting homotopy.

18 ► `GroupHomology(X,n)`
► `GroupHomology(X,n,p)`

Inputs a positive integer n and either a finite group X=G or a Coxeter diagram X=D representing an infinite Artin group G. It returns the torsion coefficients of the integral homology $H_n(G,Z)$.

There is an optional third argument which, when set equal to a prime p, causes the function to return the rank of the mod p homology $H_n(G,Zp)$.

This function is a composite of more basic functions, and makes choices for a number of parameters. For a particular group you would almost certainly be better using the more basic functions and making the choices yourself!

19 ► `HAPcopyright()`

This function provides details of HAP'S GNU public copyright licence.

20 ► `Homology(X,n)`

Inputs either a chain complex $X = C$ or a chain map $X = (C \rightarrow D)$.

If $X = C$ then the torsion coefficients of $H_n(C)$ are retuned.

If $X = (C \rightarrow D)$ then the induced homomorphism $H_n(C) \rightarrow H_n(D)$ is returned as a homomorphism of finitely presented groups.

21 ► `HomToIntegers(X)`

Inputs either a ZG-resolution $X = R$, or an equivariant chain map $X = (F : R \rightarrow S)$. It returns the cochain complex or cochain map obtained by applying $Hom_{ZG}(,Z)$ where Z is the trivial module of integers (characteristic 0).

22 ► `HomToIntegralModule(R,f)`

Inputs a ZG-resolution R and a group homomorphism $f : G \rightarrow GL_n(Z)$ to the group of $n \times n$ invertible integer matrices. Here Z must have characteristic 0. It returns the cochain complex obtained by applying $Hom_{ZG}(,A)$ where A is the ZG-module Z_n with G action via f.

23 ► `IntegralRingGenerators(R,n)`

Inputs at least n+1 terms of a ZG-resolution and integer n¿0. It returns a minimal list of cohomology classes in $H^n(G,Z)$ which, together with all cup products of lower degree classes, generate the group $H^n(G,Z)$.

(Let a_i be the i-th canonical generator of the d-generator abelian group $H^n(G,Z)$. The cohomology class $n_1 a_1 + \ldots + n_d a_d$ is represented by the integer vector $u = (n_1, \ldots, n_d)$.)

24 ► `IntegralCupProduct(R,u,v,p,q)`
► `IntegralCupProduct(R,u,v,p,q,P,Q,N)`

Inputs a ZG-resolution R, a vector u representing an element in $H^p(G,Z)$, a vector v representing an element in $H^q(G,Z)$ and the two integers $p, q > 0$. It returns a vector w representing the cup product $u \cdot v$ in $H^{p+q}(G,Z)$. This product is associative and $u \cdot v = (-1)pqv \cdot u$. It provides $H^*(G,Z)$ with the structure of an anti-commutative graded ring. The cup product is currently implemented for characteristic 0 only.

The resolution R needs a contracting homotopy.

To save the function from having to calculate the abelian groups $H^n(G,Z)$ additional input variables can be used in the form IntegralCupProduct(R,u,v,p,q,P,Q,N) , where

P is the output of the command CR_CocyclesAndCoboundaries(R,p,true)

Q is the output of the command CR_CocyclesAndCoboundaries(R,q,true)

N is the output of the command CR_CocyclesAndCoboundaries(R,p+q,true) .

25 ▶ `IsAspherical(F,R)`

Inputs a free group F and a set R of words in F. It performs a test on the 2-dimensional CW-space K associated to this presentation for the group $G = F/R^F$.

The function returns “true” if K has trivial second homotopy group. In this case it prints: Presentation is aspherical.

Otherwise it returns “fail” and prints: “Presentation is NOT piece-wise Euclidean non-positively curved”. (In this case K may or may not have trivial second homotopy group. But it is NOT possible to impose a metric on K which restricts to a Euclidean metric on each 2-cell.)

The function uses Polymake software.

26 ▶ `MultiplyWord(n,w)`

Inputs a word w and integer n. It returns the scalar multiple $n \cdot w$.

27 ▶ `Negate([i,j])`

Inputs a pair [i,j] of integers and returns [-i,j].

28 ▶ `NegateWord(w)`

Inputs a word w in a free ZG-module and returns the negated word $-w$.

29 ▶ `NonabelianExteriorProduct(G,N)`

Inputs a finite group G and normal subgroup N. It returns a record E with the following components:

E.homomorphism is a group homomorphism μ: $G \wedge N \to G$ rom the nonabelian exterior product $G \wedge N$ to G. The kernel of μ is the relative Schur multiplier.

E.pairing(x,y) is a function which inputs an element x in G and an element y in N and returns $x \wedge y$ in the exterior product $G \wedge N$.

This function should work for reasonably small nilpotent groups or extremely small non-nilpotent groups.

30 ▶ `NonabelianTensorSquare(G)`

Inputs a finite group G and returns a record T with the following components.

T.homomorphism is a group homomorphism μ: $G \otimes G \to G$ from the nonabelian tensor square of G to G. The kernel of μ is isomorphic to the third homotopy group of the suspension $SK(G,1)$ of an Eilenberg-Mac Lane space.

T.pairing(x,y) is a function which inputs two elements x, y in G and returns the tensor $x \otimes y$ in the tensor square $G \otimes G$.

This function should work for reasonably small nilpotent groups or extremely small non-nilpotent groups.

31 ▶ `PermToMatrixGroup(G,n)`

Inputs a permutation group G and its degree n. Returns a bijective homomorphism $f : G \to M$ where M is a group of permutation matrices.

32 ▶ `PolytopalComplex(G,v)`
▶ `PolytopalComplex(G,v)`

Inputs a permutation group or matrix group G of degree n and a rational vector v of length n. In both cases there is a natural action of G on v. Let P(G,v) be the convex polytope arising as the convex hull of the Euclidean points in the orbit of v under the action of G. The cellular chain complex $C_* = C_*(P(G,v))$ is an exact sequence of (not necessarily free) ZG-modules. The function returns a record R with components:

R.dimension(k) is a function which returns the number of G-orbits of the k-dimensional faces in P(G,v). If each k-face has trivial stabilizer subgroup in G then C_k is a free ZG-module of rank R.dimension(k).

R.stabilizer(k,n) is a function which returns the stabilizer subgroup for a face in the n-th orbit of k-faces.

If all faces of dimension less than k+1 have trivial stabilizer group then the first k terms of C_* constitute part of a free ZG-resolution. The boundary map is described by the function R.boundary(k,n) . (If some faces have non-trivial stabilizer group then C_* is not free and no attempt is made to determine signs for the boundary map.)

R.elements, R.group, R.properties are as in a ZG-resolution.

If an optional third input variable n is used, then only the first n terms of the resolution C_* will be computed.

The function uses Polymake software.

33 ▶ `PolytopalGenerators(G,v)`

Inputs a permutation group or matrix group G of degree n and a rational vector v of length n. In both cases there is a natural action of G on v, and the vector v must be chosen so that it has trivial stabilizer subgroup in G. Let P(G,v) be the convex polytope arising as the convex hull of the Euclidean points in the orbit of v under the action of G. The function returns a record P with components:

P.generators is a list of all those elements g in G such that gv has an edge in common with v. The list is a generating set for G.

P.vector is the vector v.

P.hasseDiagram is the Hasse diagram of the cone at v.

The function uses Polymake software. The function is joint work with Seamus Kelly.

34 ▶ `PresentationOfResolution(R)`

Inputs at least two terms of a reduced ZG-resolution R and returns a record P with components

P.freeGroup is a free group F

P.relators is a list S of words in F

where G is isomorphic to F modulo the normal closure of S. This presentation for G corresponds to the 2-skeleton of the classifying CW-space from which R was constructed. The resolution R requires no contracting homotopy.

35 ▶ `PrimePartDerivedFunctor(G,R,F,n)`

Inputs a finite group G, a positive integer n, at least n+1 terms of a ZP-resolution for a Sylow subgroup $P \leq G$ and a "mathematically suitable" covariant additive functor F such as TensorWithIntegers . It returns the abelian invariants of the p-component of the homology $H_n(F(R))$.

Warning: All calculations are assumed to be in characteristic 0. The function should not be used if the coefficient module is over the field of p elements.

"Mathematically suitable" means that the Cartan-Eilenberg double coset formula must hold.

36 ▶ `PrintZGword(w,elts)`

Inputs a word w in a free ZG-module and a (possibly partial but sufficient) listing elts of the elements of G. The function prints the word w to the screen in the form

$$r_1E_1 + \ldots + r_nE_n$$

where r_i are elements in the group ring ZG, and E_i denotes the i-th free generator of the module.

37 ▶ `RelativeSchurMultiplierG,N)`

Inputs a finite group G and normal subgroup N. It returns the homology group $H_2(G,N,Z)$ that fits into the exact sequence

$$H_3(G,Z) \to H_3(G/N,Z) \to H_2(G,N,Z) \to H_2(G,Z) \to H_2(G/N,Z)$$

This function should work for reasonably small nilpotent groups G or extremely small non-nilpotent groups.

38 ► `ResolutionAbelianGroup(L,n)`
► `ResolutionAbelianGroup(G,n)`

Inputs a list $L := [m_1, m_2, \ldots, m_d]$ of nonnegative integers, and a positive integer n. It returns n terms of a ZG-resolution for the abelian group $G = Z_{L[1]} \oplus Z_{L[2]} \oplus \cdots \oplus Z_{L[d]}$.

The first argument can also be the abelian group G itself.

39 ► `ResolutionArtinGroup(D,n)`

Inputs a Coxeter diagram D and an integer n¿1. It returns n terms of a free ZG-resolution R where G is the Artin monoid associated to D. It is conjectured that R is also a free resolution for the Artin group G. The conjecture is known to hold in certain cases.

G=R.group is infinite and returned as a finitely presented group. The list R.elts is a partial listing of the elements of G which grows as R is used. Initially R.elts is empty and then, any time the boundary of a resolution generator is called, R.elts is updated to include elements of G involved in the boundary.

The contracting homotopy on R has not yet been implemented!

40 ► `ResolutionAsphericalPresentation(F,R,n)`

Inputs a free group F, a set R of words in F which constitute an aspherical presentation for a group G, and a positive integer n. (Asphericity can be a difficult property to verify. The function IsAspherical(F,R) could be of help.)

The function returns n terms of a free ZG-resolution R which has generators in dimensions ≤ 2 only. No contracting homotopy on R will be returned.

41 ► `ResolutionDirectProduct(R,S)`

Inputs a ZG-resolution R and ZH-resolution S. It outputs a ZD-resolution for the direct product $D = G \times H$.

42 ► `ResolutionFiniteExtension(gensE,gensG,R,n)`
► `ResolutionFiniteExtension(gensE,gensG,R,n,true)`
► `ResolutionFiniteExtension(gensE,gensG,R,n,false,S)`

Inputs: a set gensE of generators for a finite group E; a set gensG equal to the image of gensE in a quotient group G of E; a ZG-resolution R up to dimension at least n; a positive integer n. It uses the TwistedTensorProduct(construction to return n terms of a ZE-resolution.

The function has an optional fourth argument which, when set equal to true, invokes tietze reductions in the construction of a resolution for the kernel of $E \to G$.

If a ZN-resolution S is available, where N is the kernel of the quotient $E \to G$, then this can be incorporated into the computations using an optional fifth argument.

The contracting homotopy on the ZE-resolution has not yet been implemented!

43 ► `ResolutionFiniteGroup(gens,n)`
► `ResolutionFiniteGroup(gens,n,true)`
► `ResolutionFiniteGroup(gens,n,false,p)`

Inputs a set gens of generators for a finite group G and a positive integer n. It outputs n terms of a ZG-resolution.

The function has an optional third argument which, when set equal to true, invokes tietze reductions in the construction of the resolution.

The function has an optional fourth argument which, when set equal to a prime p, records the fact that the resolution will only be used for mod p calculations. This could speed up subsequent constructions.

44 ▶ `ResolutionFiniteSubgroup(R,K)`
▶ `ResolutionFiniteSubgroup(R,gensG,gensK)`

Inputs a ZG-resolution for a finite group G and a subgroup K of index G:K. It returns a free ZK-resolution whose ZK-rank is G:K times the ZG-rank in each dimension.

Generating sets gensG, gensK for G and K can also be input to the function (though the method does not depend on a choice of generators).

This ZK-resolution has more than one generator in dimension 0. So PresentationOfResolution() should not be applied to it!

45 ▶ `ResolutionNormalSeries(L,n)`
▶ `ResolutionNormalSeries(L,n,true)`
▶ `ResolutionNormalSeries(L,n,false,p)`

Inputs a positive integer n and a list $L = [L_1, \ldots, L_k]$ of normal subgroups L_i of a finite group G satisfying $G = L1 \geq L_2 \ldots \geq L_k$. Alternatively, $L = [gensL_1, \ldots gensL_k]$ can be a list of generating sets for the L_i (and these particular generators will be used in the construction of resolutions). It returns a $Z(G/L_k)$-resolution by repeatedly using the function ResolutionOfFiniteExtension( ). Typically $L_k = 1$ in which case a ZG-resolution is returned.

The function has an optional third argument which, if set equal to “true”, invokes tietze reductions in the construction of resolutions.

The function has an optional fourth argument which, if set equal to p¿0, produces a resolution which is only valid for mod p calculations.

The contracting homotopy on the ZG-resolution has not yet been implemented!

46 ▶ `ResolutionPrimePowerGroup(G,n)`

Inputs a p-group G and integer $n \geq 1$. It uses GAP’s standard linear algebra functions over the field Z_p to construct a ZG-resolution for mod p calculations only. The resolution is probably minimal - meaning that the number of generators of R_n equals the rank of $H_n(G, Z_p)$. However, the implementation takes a “short cut” and so I don’t think minimality can be guaranteed.

The contracting homotopy on the ZG-resolution has not yet been implemented!

47 ▶ `ResolutionSmallFpGroup(G,n)`
▶ `ResolutionSmallFpGroup(G,n,p)`

Inputs a small finitely presented group G and an integer n¿0. It returns n terms of a ZG-resolution which, in dimensions 1 and 2, corresponds to the given presentation for G. The method returns no contracting homotopy for the resolution.

The function has an optional fourth argument which, when set equal to a prime p, records the fact that the resolution will only be used for mod p calculations. This could speed up subsequent constructions.

This function was written by Irina Kholodna.

48 ▶ `ResolutionSubgroup(R,K)`

Inputs a ZG-resolution for an (infinite) group G and a subgroup K of finite index G:K. It returns a free ZK-resolution whose ZK-rank is G:K times the ZG-rank in each dimension.

If G is finite then the function ResolutionFiniteSubgroup(R,G,K) will probably work better. In particular, resolutions from this function probably won’t work with the function EquivariantChainMap(.

This ZK-resolution has more than one generator in dimension 0. So PresentationOfResolution() should not be applied to it!

49 ► `StandardCocycle(R,f,n)`
► `StandardCocycle(R,f,n,q)`

Inputs a ZG-resolution R (with contracting homotopy), a positive integer n and an integer vector f representing an n-cocycle $R_n \to Z_q$ where G acts trivially on Z_q. It is assumed $q = 0$ unless a value for q is entered. The command returns a function $F(g_1, \ldots, g_n)$ which is the standard cocycle $G_n \to Z_q$ corresponding to f. At present the command is implemented only for n=2 or 3.

50 ► `Syzygy(R,g)`

Inputs a ZG-resolution R (with contracting homotopy) and a list $g = [g[1], \ldots, g[n]]$ of elements in G. It returns a word w in R_n. The word w is the image of the n-simplex in the standard bar resolution corresponding to the n-tuple g. This function can be used to construct explicit standard n-cocycles. (Currently implemented only for $n \leq 3$.)

51 ► `TensorWithIntegers(X)`

Inputs either a ZG-resolution $X = R$, or an equivariant chain map $X = (F : R \to S)$. It returns the chain complex or chain map obtained by tensoring with the trivial module of integers (characteristic 0).

52 ► `TensorWithIntegersModP(X,p)`

Inputs either a ZG-resolution $X = R$, or an equivariant chain map $X = (F : R \to S)$, and a prime p. It returns the chain complex or chain map obtained by tensoring with the trivial module of integers modulo p.

53 ► `ThirdHomotopyGroupOfSuspensionB(G)`

Inputs a finite group G and returns the abelian invariants of the third homotopy group of the suspension SB(G) of the Eilenberg-Mac Lane space K(G,1). This function should work for reasonably small nilpotent groups or extremely small non-nilpotent groups.

54 ► `TietzeReduction(S,w)`

Inputs a set S of words in a free ZG-module, and a word w in the module. The function returns a word w' such that $\{S, w'\}$ generates the same abelian group as $\{S, w\}$. The word w' is possibly shorter (and certainly no longer) than w. This function needs to be improved!

55 ► `TorsionGeneratorsAbelianGroup(G)`

Inputs an abelian group G and returns a generating set $[x_1, \ldots, x_n]$ where no pair of generators have coprime orders.

56 ► `TwistedTensorProduct(R,S,EhomG,GmapE,NhomE,NEhomN,EltsE,Mult,InvE)`

Inputs a ZG-resolution R, a ZN-resolution S, and other data relating to a short exact sequence $1 \to N \to E \to G \to 1$. It uses a perturbation technique of CTC Wall to construct a ZE-resolution F. Both G and N could be infinite. The "length" of F is equal to the minimum of the "length"s of R and S. The resolution R needs no contracting homotopy if no such homotopy is requied for F.

The contracting homotopy on F has not yet been implemented!

57 ► `VectorStabilizer(G,v)`

Inputs a permutation group or matrix group G of degree n and a rational vector of degree n. In both cases there is a natural action of G on v and the function returns the group of elements in G that fix v.

Index

This index covers only this manual. A page number in *italics* refers to a whole section which is devoted to the indexed subject. Keywords are sorted with case and spaces ignored, e.g., "PermutationCharacter" comes before "permutation group".