

Contents

1	Resolutions of the ground ring	2
2	Resolutions of modules	3
3	Induced equivariant chain maps	4
4	Functors	5
5	Chain complexes	6
6	Homology and cohomology groups	7
7	Poincare series	8
8	Cohomology ring structure	9
9	Commutator and nonabelian tensor computations	10
10	Generators and relators of groups	11
11	Orbit polytopes and fundamental domains	12
12	Cocycles	14
13	Words in free ZG-modules	15
14	FpG-modules	16
15	Meataxe modules	17
16	Coxeter diagrams and graphs of groups	18
17	Some functions for accessing basic data	19
18	Miscellaneous	20

Chapter 1

Resolutions of the ground ring

`ResolutionAbelianGroup(L,n)` `ResolutionAbelianGroup(G,n)` Inputs a list $L := [m_1, m_2, \dots, m_d]$ of nonnegative integers and a positive integer n . It returns the resolution of the group defined by L and n .

`ResolutionAlmostCrystalGroup(G,n)` Inputs a positive integer n and an almost crystallographic pcg group G . It returns the resolution of the group defined by G and n .

`ResolutionAlmostCrystalQuotient(G,n,c)` `ResolutionAlmostCrystalQuotient(G,n,c,false)` An almost crystallographic pcg group G and a positive integer n . It returns the resolution of the group defined by G and n .

`ResolutionArtinGroup(D,n)` Inputs a Coxeter diagram D and an integer $n > 1$. It returns n terms of a free ZG -resolution of the Artin group defined by D and n .

`ResolutionAsphericalPresentation(F,R,n)` Inputs a free group F , a set R of words in F which constitute an aspherical presentation of a group, and a positive integer n . It returns the resolution of the group defined by F and R .

`ResolutionBieberbachGroup(G)` `ResolutionBieberbachGroup(G,v)` Inputs a Bieberbach group G (represented using `AffineCrystGroupOnRight`) and a positive integer v . It returns the resolution of the group defined by G and v .

`ResolutionDirectProduct(R,S)` `ResolutionDirectProduct(R,S,"internal")` Inputs a ZG -resolution R and a ZG -resolution S . It returns the resolution of the direct product of the groups defined by R and S .

`ResolutionExtension(g,R,S)` `ResolutionExtension(g,R,S,"TestFiniteness")` `ResolutionExtension(g,R,S,"TestFiniteness",true)` Inputs a group g , a ZG -resolution R , and a ZG -resolution S . It returns the resolution of the extension of the group defined by R by the group defined by S .

`ResolutionFiniteDirectProduct(R,S)` `ResolutionFiniteDirectProduct(R,S,"internal")` Inputs a ZG -resolution R and a ZG -resolution S . It returns the resolution of the direct product of the groups defined by R and S .

`ResolutionFiniteExtension(gensE,gensG,R,n)` `ResolutionFiniteExtension(gensE,gensG,R,n,true)` Inputs a list $gensE$ of generators of a group E , a list $gensG$ of generators of a group G , a ZG -resolution R , and a positive integer n . It returns the resolution of the extension of the group defined by R by the group defined by $gensE$ and $gensG$.

`ResolutionFiniteGroup(gens,n)` `ResolutionFiniteGroup(gens,n,true)` `ResolutionFiniteGroup(gens,n,true,true)` Inputs a list $gens$ of generators of a group G and a positive integer n . It returns the resolution of the group defined by $gens$ and n .

`ResolutionFiniteSubgroup(R,K)` `ResolutionFiniteSubgroup(R,gensG,gensK)` Inputs a ZG -resolution R for a group G and a list $gensK$ of generators of a subgroup K of G . It returns the resolution of the subgroup defined by R and $gensK$.

`ResolutionGraphOfGroups(D,n)` `ResolutionGraphOfGroups(D,n,L)` Inputs a graph of groups D and a positive integer n . It returns the resolution of the group defined by D and n .

`ResolutionNilpotentGroup(G,n)` `ResolutionNilpotentGroup(G,n,"TestFiniteness")` Inputs a nilpotent group G and a positive integer n . It returns the resolution of the group defined by G and n .

`ResolutionNormalSeries(L,n)` `ResolutionNormalSeries(L,n,true)` `ResolutionNormalSeries(L,n,false)` Inputs a list L of subgroups of a group G and a positive integer n . It returns the resolution of the group defined by L and n .

`ResolutionPrimePowerGroup(P,n)` `ResolutionPrimePowerGroup(G,n,p)` Inputs a p -group P and integer $n > 0$. It returns the resolution of the group defined by P and n .

`ResolutionSmallFpGroup(G,n)` `ResolutionSmallFpGroup(G,n,p)` Inputs a small finitely presented group G and integer $n > 0$. It returns the resolution of the group defined by G and n .

`ResolutionSubgroup(R,K)` Inputs a ZG -resolution for an (infinite) group G and a subgroup K of finite index $|G : K|$. It returns the resolution of the subgroup defined by R and K .

`ResolutionSubnormalSeries(L,n)` Inputs a positive integer n and a list $L = [L_1, \dots, L_k]$ of subgroups L_i of a finite group G . It returns the resolution of the group defined by L and n .

`TwistedTensorProduct(R,S,EhomG,GmapE,NhomE,NEhomN,EltSE,Mult,InvE)` Inputs a ZG -resolution R , a ZN -resolution S , and a list of maps $EhomG, GmapE, NhomE, NEhomN, EltSE, Mult, InvE$ defining a twisted tensor product. It returns the resolution of the group defined by R and S .

Chapter 2

Resolutions of modules

| `ResolutionFpGModule(M, n)` Inputs an FpG -module M and a positive integer n . It returns n terms of a minimal free F

Chapter 3

Induced equivariant chain maps

| `EquivariantChainMap(R, S, f)` Inputs a ZG -resolution R , a ZG' -resolution S , and a group homomorphism $f : G \longrightarrow G'$

Chapter 4

Functors

•

`HomToIntegers(X)` Inputs either a ZG -resolution $X = R$, or an equivariant chain map $X = (F : R \longrightarrow S)$. It returns the

`HomToIntegersModP(R)` Inputs a ZG -resolution R and returns the cochain complex obtained by applying $Hom_{ZG}(Z, R)$ to

`HomToIntegralModule(R, f)` Inputs a ZG -resolution R and a group homomorphism $f : G \longrightarrow GL_n(Z)$ to the group of

`LowerCentralSeriesLieAlgebra(G)` `LowerCentralSeriesLieAlgebra(f)` Inputs a pcg group G . If each quotient G_i/G_{i+1} is

`TensorWithIntegers(X)` Inputs either a ZG -resolution $X = R$, or an equivariant chain map $X = (F : R \longrightarrow S)$. It returns

`TensorWithIntegersModP(X, p)` Inputs either a ZG -resolution $X = R$, or an equivariant chain map $X = (F : R \longrightarrow S)$. It returns

`TensorWithRationals(R)` Inputs a ZG -resolution R and returns the chain complex obtained by tensoring with the trivial

Chapter 5

Chain complexes

`ChevalleyEilenbergComplex(X, n)` Inputs either a Lie algebra $X = A$ (over the ring of integers Z or over a field K) or

`LeibnizComplex(X, n)` Inputs either a Lie or Leibniz algebra $X = A$ (over the ring of integers Z or over a field K) or

Chapter 6

Homology and cohomology groups

`Cohomology(X)` Inputs either a cochain complex $X = C$ or a cochain map $X = (C \longrightarrow D)$ over the integers Z . If $X = C$ then the torsion coefficients are computed.
`GroupCohomology(X,n)` `GroupCohomology(X,n,p)` Inputs a positive integer n and either a finite group $X = G$ or a Coxeter diagram $X = D$ representing a finite group.
`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 a finite group.
`Homology(X,n)` Inputs either a chain complex $X = C$ or a chain map $X = (C \longrightarrow D)$. If $X = C$ then the torsion coefficients are computed.
`HomologyPb(C,n)` This is a back-up function which might work in some instances where `Homology(C,n)` fails. It is more computationally intensive.
`LeibnizAlgebraHomology(A,n)` Inputs a Lie or Leibniz algebra $X = A$ (over the ring of integers Z or over a field K), and a positive integer n . It returns the n -th homology group.
`LieAlgebraHomology(A,n)` Inputs a Lie algebra A (over the integers or a field) and a positive integer n . It returns the n -th homology group.
`PrimePartDerivedFunctor(G,R,F,n)` Inputs a finite group G , a positive integer n , at least $n+1$ terms of a ZP -resolution of G over R , and a field F . It returns the n -th derived functor.
`RankHomologyPGroup(G,n)` `RankHomologyPGroup(R,n)` `RankHomologyPGroup(G,n,"empirical")` Inputs a (smallish) p -group G together with a positive integer n . It returns a function $dim(k)$ for $k = 0, 1, \dots, n$.
`RankPrimeHomology(G,n)` Inputs a (smallish) p -group G together with a positive integer n . It returns a function $dim(k)$ for $k = 0, 1, \dots, n$.

Chapter 7

Poincare series

•

`EfficientNormalSubgroups (G)`

`EfficientNormalSubgroups (G, k)` Inputs a prime-power group G and, optionally, a positive integer k . The default is $k = 1$.

`ExpansionOfRationalFunction (f, n)` Inputs a positive integer n and a rational function $f(x) = p(x)/q(x)$ where the coefficients are in a field.

`PoincareSeries (G, n)` `PoincareSeries (R, n)`

`PoincareSeries (L, n)`

`PoincareSeries (G)` Inputs a finite p -group G and a positive integer n . It returns a quotient of polynomials $f(x) = P_n(x)/Q_n(x)$.

`PoincareSeriesPrimePart (G, p, n)` Inputs a finite group G , a prime p , and a positive integer n . It returns a quotient of polynomials.

`Prank (G)` Inputs a p -group G and returns the rank of the largest elementary abelian subgroup.

Chapter 8

Cohomology ring structure

`IntegralCupProduct (R, u, v, p, q)`

`IntegralCupProduct (R, u, v, p, q, P, Q, N)` (Various functions used to construct the cup product are also *CRfunctions*)

`IntegralRingGenerators (R, n)` Inputs at least $n + 1$ terms of a ZG -resolution and integer $n > 0$. It returns a minimal

`ModPCohomologyRing (G, n)`

`ModPCohomologyRing (R)` Inputs either a p -group G and positive integer n , or else n terms of a minimal $Z_p G$ -resolution

`ModP RingGenerators (A)` Inputs a mod p cohomology ring A (created using the preceding function). It returns a gen

Chapter 9

Commutator and nonabelian tensor computations

•

`BaerInvariant(G, c)` Inputs a nilpotent group G and integer $c > 0$. It returns the Baer invariant $M^{(c)}(G)$ defined as follows.

`Coclass(G)` Inputs a group G of prime-power order p^n and nilpotency class c say. It returns the integer $r = n - c$.

`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

`NonabelianExteriorProduct(G, N)` Inputs a finite group G and normal subgroup N . It returns a record E with the following

`NonabelianTensorProduct(G, N)` Inputs a finite group G and normal subgroup N . It returns a record E with the following

`NonabelianTensorSquare(G)`

`NonabelianTensorSquare(G, m)` Inputs a finite or nilpotent infinite group G and returns a record T with the following

`RelativeSchurMultiplier(G, N)` Inputs a finite group G and normal subgroup N . It returns the homology group $H_2(N, Z(G))$.

`TensorCentre(G)` Inputs a group G and returns the largest central subgroup N such that the induced homomorphism $G/N \rightarrow G/N$ is

`ThirdHomotopyGroupOfSuspensionB(G)`

`ThirdHomotopyGroupOfSuspensionB(G, m)` Inputs a finite or nilpotent infinite group G and returns the abelian invariant $\gamma_3(G)$.

`UpperEpicentralSeries(G, c)` Inputs a nilpotent group G and an integer c . It returns the c -th term of the upper epicentral series of G .

Chapter 10

Generators and relators of groups

•

`CayleyGraphDisplay(G,X)`

`CayleyGraphDisplay(G,X,"mozilla")` Inputs a finite group G together with a subset X of G . It displays the corresponding Cayley graph.

`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 defined by (F,R) .

`PresentationOfResolution(R)` Inputs at least two terms of a reduced ZG -resolution R and returns a record P with the presentation of $H_2(G)$.

`TorsionGeneratorsAbelianGroup(G)` Inputs an abelian group G and returns a generating set $[x_1, \dots, x_n]$ where no proper subset of $\{x_1, \dots, x_n\}$ generates G .

Chapter 11

Orbit polytopes and fundamental domains

•
`FundamentalDomainAffineCrystGroupOnRight(v, G)` Inputs a crystallographic group G (represented using `AffineC`
`OrbitPolytope(G, v, L)` Inputs a permutation group or matrix group G of degree n and a rational vector v of length n .
The function uses Polymake software.

`PolytopalComplex(G, v)`
`PolytopalComplex(G, v, n)`

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 component object 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 $< k + 1$ 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 $boundary(k, n)$. (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.

`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 $g \cdot v$ 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.

`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 .

Chapter 12

Cocycles

`CocycleCondition(R, n)` Inputs a resolution R and an integer $n > 0$. It returns an integer matrix M with the following
`StandardCocycle(R, f, n)`
`StandardCocycle(R, f, n, q)` Inputs a ZG -resolution R (with contracting homotopy), a positive integer n and an integer q .
`Syzygy(R, g)` Inputs a ZG -resolution R (with contracting homotopy) and a list $g = [g[1], \dots, g[n]]$ of elements in G . It returns

Chapter 13

Words in free ZG -modules

`AddFreeWords(v, w)` Inputs two words v, w in a free ZG -module and returns their sum $v + w$. If the characteristic of Z is p , then $v + w$ is reduced modulo p .

`AddFreeWordsModP(v, w, p)` Inputs two words v, w in a free ZG -module and the characteristic p of Z . It returns the sum $v + w$ reduced modulo p .

`AlgebraicReduction(w)` Inputs a word w in a free ZG -module and returns a reduced version of the word in which no subword is a power of a generator.

`AlgebraicReduction(w, p)` Inputs a word w in a free ZG -module and returns a reduced version of the word in which no subword is a power of a generator, and the result is reduced modulo p .

`Multiply Word(n, w)` Inputs a word w and integer n . It returns the scalar multiple $n \cdot w$.

`Negate([i, j])` Inputs a pair $[i, j]$ of integers and returns $[-i, j]$.

`NegateWord(w)` Inputs a word w in a free ZG -module and returns the negated word $-w$.

`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 . It prints the word w in terms of the elements of G .

`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 reduced version of w in which no subword is in S .

Chapter 14

FpG -modules

`DirectSumOfFpGModules (M, N)`
`DirectSumOfFpGModules ([M[1], M[2], ..., M[k]])` Inputs two FpG -modules M and N with common group G .
`FpGModule (A, P)` Inputs a p -group P and a matrix A whose rows have length a multiple of the order of G . It returns an FpG -module.
`FpGModuleDualBasis (M)` Inputs an FpG -module M . It returns a record R with two components: $R.freeModule$ is the free module and $R.dualBasis$ is a dual basis.
`FpGModuleHomomorphism (M, N, A)` Inputs FpG -modules M and N over a common p -group G . Also inputs a list A of matrices.
`DesuspensionFpGModule (M, n)` Inputs a positive integer n and an FpG -module M . It returns an FpG -module $D^n M$.
`RadicalOfFpGModule (M)` Inputs an FpG -module M with G a p -group, and returns the Radical of M as an FpG -module.
`GeneratorsOfFpGModule (M)` Inputs an FpG -module M and returns a matrix whose rows correspond to a minimal generating set.
`ImageOfFpGModuleHomomorphism (f)` Inputs an FpG -module homomorphism $f : M \rightarrow N$ and returns its image $f(M)$.
`IntersectionOfFpGModules (M, N)` Inputs two FpG -modules M, N arising as submodules in a common free module.
`IsFpGModuleHomomorphismData (M, N, A)` Inputs FpG -modules M and N over a common p -group G . Also inputs a list A of matrices.
`MultipleOfFpGModule (w, M)` Inputs an FpG -module M and a list $w := [g_1, \dots, g_t]$ of elements in the group $G = M! : g$.
`ProjectedFpGModule (M, k)` Inputs an FpG -module M of ambient dimension $n|G|$, and an integer k between 1 and n .
`RandomHomomorphismOfFpGModules (M, N)` Inputs two FpG -modules M and N over a common group G . It returns a random homomorphism.
`Rank (f)` Inputs an FpG -module homomorphism $f : M \rightarrow N$ and returns the dimension of the image of f as a vector space.
`SumOfFpGModules (M, N)` Inputs two FpG -modules M, N arising as submodules in a common free module $(FG)^n$ where n is the ambient dimension.
`SumOp (f, g)` Inputs two FpG -module homomorphisms $f, g : M \rightarrow N$ with common source and common target. It returns the sum $f + g$.
`VectorsToFpGModuleWords (M, L)` Inputs an FpG -module M and a list $L = [v_1, \dots, v_k]$ of vectors in M . It returns a list of words.

Chapter 15

Meataxe modules

•
| DesuspensionMtxModule(M) Inputs a meataxe module M over the field of p elements and returns an FpG-module DM .
| GeneratorsOfMtxModule(M) Inputs a meataxe module M acting on, say, the vector space V . The function returns a m

Chapter 16

Coxeter diagrams and graphs of groups

`CoxeterDiagramComponents(D)` Inputs a Coxeter diagram D and returns a list $[D_1, \dots, D_d]$ of the maximal connected components of D .

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

`CoxeterDiagramDisplay(D)` Inputs a Coxeter diagram D and displays it as a .gif file. It uses the `GraphOfGroupsDisplay` function.

`CoxeterDiagramDisplay(D, "web browser")` Inputs a Coxeter diagram D and displays it as a .gif file. It uses the `GraphOfGroupsDisplay` function.

`CoxeterDiagramFpArtinGroup(D)` Inputs a Coxeter diagram D and returns the corresponding finitely presented Artin group.

`CoxeterDiagramFpCoxeterGroup(D)` Inputs a Coxeter diagram D and returns the corresponding finitely presented Coxeter group.

`CoxeterDiagramIsSpherical(D)` Inputs a Coxeter diagram D and returns "true" if the associated Coxeter groups is spherical.

`CoxeterDiagramMatrix(D)` Inputs a Coxeter diagram D and returns a matrix representation of it. The matrix is given by (m_{ij}) where m_{ij} is the order of $s_i s_j$.

`CoxeterSubDiagram(D, V)` Inputs a Coxeter diagram D and a subset V of its vertices. It returns the full sub-diagram of D with vertices V .

`CoxeterDiagramVertices(D)` Inputs a Coxeter diagram D and returns its set of vertices.

`EvenSubgroup(G)` Inputs a group G and returns a subgroup G^+ . The subgroup is that generated by all products xy where x, y are in G .

`GraphOfGroupsDisplay(D)` Inputs a graph of groups D and displays it as a .gif file. It uses the `GraphOfGroupsDisplay` function.

`GraphOfGroupsDisplay(D, "web browser")` Inputs a graph of groups D and displays it as a .gif file. It uses the `GraphOfGroupsDisplay` function.

`GraphOfGroupsTest(D)` Inputs an object D and tries to test whether it is a Graph of Groups. However, it DOES NOT work.

Chapter 17

Some functions for accessing basic data

`BoundaryMap(C)` Inputs a resolution, chain complex or cochain complex C and returns the function $C!.boundary$.

`BoundaryMatrix(C,n)` Inputs a chain or cochain complex C and integer $n>0$. It returns the n -th boundary map of C .

`Dimension(C)`

`Dimension(M)` Inputs a resolution, chain complex or cochain complex C and returns the function $C!.dimension$. Alternatively, `Dimension(M)` returns the dimension of the resolution M .

`EvaluateProperty(X,"name")` Inputs a component object X (such as a ZG -resolution or chain map) and a string "name". It returns the value of the property "name" of X .

`GroupOfResolution(R)` Inputs a ZG -resolution R and returns the group G .

`Length(R)` Inputs a resolution R and returns its length (i.e. the number of terms of R that HAP has computed).

`Map(f)` Inputs a chain map, or cochain map or equivariant chain map f and returns the mapping function (as opposed to the mapping matrix).

`Source(f)` Inputs a chain map, or cochain map, or equivariant chain map, or FpG -module homomorphism f and returns the source module.

`Target(f)` Inputs a chain map, or cochain map, or equivariant chain map, or FpG -module homomorphism f and returns the target module.

Chapter 18

Miscellaneous

`BigStepLCS(G, n)` Inputs a group G and a positive integer n . It returns a subseries $G = L_1 > L_2 > \dots L_k = 1$ of the lower central series of G .

`Compose(f, g)` Inputs two FpG -module homomorphisms $f : M \longrightarrow N$ and $g : L \longrightarrow M$ with $Source(f) = Target(g)$.

`HAPcopyright()` This function provides details of HAP'S GNU public copyright licence.

`IsLieAlgebraHomomorphism(f)` Inputs an object f and returns true if f is a homomorphism $f : A \longrightarrow B$ of Lie algebras.

`IsSuperperfect(G)` Inputs a group G and returns "true" if both the first and second integral homology of G is trivial.

`MakeHAPManual()` This function creates the manual for HAP from an XML file.

`PermToMatrixGroup(G, n)` Inputs a permutation group G and its degree n . Returns a bijective homomorphism $f : G \longrightarrow GL(n, F)$.

`SolutionsMatDestructive(M, B)` Inputs an $m \times n$ matrix M and a $k \times n$ matrix B over a field. It returns a $k \times m$ matrix C such that $CB = M$.

`TestHap()` This runs a representative sample of HAP functions and checks to see that they produce the correct output.

Index

- AddFreeWords, [15](#)
- AddFreeWordsModP, [15](#)
- AlgebraicReduction, [15](#)

- BaerInvariant, [10](#)
- BigStepLCS, [20](#)
- BoundaryMap, [19](#)
- BoundaryMatrix, [19](#)

- CayleyGraphDisplay, [11](#)
- ChevalleyEilenbergComplex, [6](#)
- Coclass, [10](#)
- CocycleCondition, [14](#)
- Cohomology, [7](#)
- Compose(f,g), [20](#)
- CoxeterDiagramComponents, [18](#)
- CoxeterDiagramDegree, [18](#)
- CoxeterDiagramDisplay, [18](#)
- CoxeterDiagramFpArtinGroup, [18](#)
- CoxeterDiagramFpCoxeterGroup, [18](#)
- CoxeterDiagramIsSpherical, [18](#)
- CoxeterDiagramMatrix, [18](#)
- CoxeterDiagramVertices, [18](#)
- CoxeterSubDiagram, [18](#)

- DesuspensionFpGModule, [16](#)
- DesuspensionMtxModule, [17](#)
- Dimension, [19](#)
- DirectSumOfFpGModules, [16](#)

- EpiCentre, [10](#)
- EquivariantChainMap, [4](#)
- EvaluateProperty, [19](#)
- EvenSubgroup, [18](#)
- ExpansionOfRationalFunction, [8](#)

- FpGModule, [16](#)
- FpGModuleDualBasis, [16](#)
- FpGModuleHomomorphism, [16](#)
- Fundamental domains (HAPcryst), [12](#)

- GeneratorsOfFpGModule, [16](#)
- GeneratorsOfMtxModule, [17](#)
- GraphOfGroupsDisplay, [18](#)
- GraphOfGroupsTest, [18](#)
- GroupCohomology, [7](#)
- GroupHomology, [7](#)
- GroupOfResolution, [19](#)

- HAPcopyright, [20](#)
- Homology, [7](#)
- HomologyPb, [7](#)
- HomToIntegers, [5](#)
- HomToIntegersModP, [5](#)
- HomToIntegralModule, [5](#)

- ImageOfFpGModuleHomomorphism, [16](#)
- IntegralCupProduct, [9](#)
- IntegralRingGenerators, [9](#)
- IntersectionOfFpGModules, [16](#)
- IsAspherical, [11](#)
- IsFpGModuleHomomorphismData, [16](#)
- IsLieAlgebraHomomorphism, [20](#)
- IsSuperperfect, [20](#)

- LeibnizAlgebraHomology, [7](#)
- LeibnizComplex, [6](#)
- Length, [19](#)
- LieAlgebraHomology, [7](#)
- LowerCentralSeriesLieAlgebra, [5](#)

- MakeHAPManual, [20](#)
- Map, [19](#)
- ModPCohomologyRing, [9](#)
- ModP RingGenerators, [9](#)
- MultipleOfFpGModule, [16](#)
- MultiplyWord, [15](#)

- Negate, [15](#)
- NegateWord, [15](#)
- NonabelianExteriorProduct, [10](#)

NonabelianTensorProduct, 10
 NonabelianTensorSquare, 10
 OrbitPolytope, 12
 PermToMatrixGroup, 20
 PoincareSeries, 8
 PoincareSeriesPrimePart, 8
 PolytopalComplex, 12
 PolytopalGenerators, 12
 Prank, 8
 PresentationOfResolution, 11
 PrimePartDerivedFunctor, 7
 PrintZGword, 15
 ProjectedFpGModule, 16
 RadicalOfFpGModule, 16
 RandomHomomorphismOfFpGModules, 16
 Rank, 16
 RankHomologyPGroup, 7
 RankPrimeHomology, 7
 RelativeSchurMultiplier, 10
 ResolutionAbelianGroup, 2
 ResolutionAlmostCrystalGroup, 2
 ResolutionAlmostCrystalQuotient, 2
 ResolutionArtinGroup, 2
 ResolutionAsphericalPresentation, 2
 ResolutionBieberbachGroup (HAPcryst), 2
 ResolutionDirectProduct, 2
 ResolutionExtension, 2
 ResolutionFiniteDirectProduct, 2
 ResolutionFiniteExtension, 2
 ResolutionFiniteGroup, 2
 ResolutionFiniteSubgroup, 2
 ResolutionFpGModule, 3
 ResolutionGraphOfGroups, 2
 ResolutionNilpotentGroup, 2
 ResolutionNormalSeries, 2
 ResolutionPrimePowerGroup, 2
 ResolutionSmallFpGroup, 2
 ResolutionSubgroup, 2
 ResolutionSubnormalSeries, 2
 SolutionsMatDestructive, 20
 Source, 19
 StandardCocycle, 14
 SumOfFpGModules, 16
 SumOp, 16
 Syzygy, 14
 Target, 19
 TensorCentre, 10
 TensorWithIntegers, 5
 TensorWithIntegersModP, 5
 TensorWithRationals, 5
 TestHap, 20
 ThirdHomotopyGroupOfSuspensionB, 10
 TietzeReduction, 15
 TorsionGeneratorsAbelianGroup, 11
 TwistedTensorProduct, 2
 UpperEpicentralSeries, 10
 VectorStabilizer, 13
 VectorsToFpGModuleWords, 16