

Contents

1	Resolutions of the ground ring	3
2	Resolutions of modules	4
3	Induced equivariant chain maps	5
4	Functors	6
5	Chain complexes	7
6	Homology and cohomology groups	8
7	Poincare series	9
8	Cohomology ring structure	10
9	Cohomology rings of p-groups (mainly $p = 2$)	11
10	Commutator and nonabelian tensor computations	12
11	Lie commutators and nonabelian Lie tensors	13
12	Generators and relators of groups	14
13	Orbit polytopes and fundamental domains	15
14	Cocycles	16
15	Words in free ZG-modules	17
16	FpG-modules	18
17	Meataxe modules	19
18	G-Outer Groups	20
19	Cat-1-groups	21
20	Coxeter diagrams and graphs of groups	22

	2
21 Simplicial Complexes	23
22 Cubical Complexes	24
23 Commutative diagrams and abstract categories	25
24 Arrays and Pseudo lists	26
25 Parallel Computation - Core Functions	27
26 Parallel Computation - Extra Functions	28
27 Some functions for accessing basic data	29
28 Miscellaneous	30

Chapter 1

Resolutions of the ground ring

TietzeReducedResolution(R) Inputs a $\mathbb{Z}G$ -resolution R and returns a $\mathbb{Z}G$ -resolution S which is obtained from R by Tietze transformations.

FreeGResolution(P, n) FreeGResolution(P, n, p) Inputs a non-free ZG -resolution P with finite stabilizer group G and an integer n . The second version also takes a prime p .

ResolutionAbelianGroup(L, n) ResolutionAbelianGroup(G, n) Inputs a list $L := [m_1, m_2, \dots, m_d]$ of nonnegative integers and an integer n . The second version takes a group G .

ResolutionAlmostCrystalGroup(G, n) Inputs a positive integer n and an almost crystallographic pcg group G . It returns a resolution of $\mathbb{Z}G$.

ResolutionAlmostCrystalQuotient(G, n, c) ResolutionAlmostCrystalQuotient(G, n, c, false) An algorithm to compute a resolution of $\mathbb{Z}G$ for a group G which is a quotient of an almost crystallographic pcg group by a normal subgroup of index c .

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

ResolutionAsphericalPresentation(F, R, n) Inputs a free group F , a set R of words in F which constitute an aspherical presentation of a group G , and an integer n . It returns a resolution of $\mathbb{Z}G$.

ResolutionBieberbachGroup(G) ResolutionBieberbachGroup(G, v) Inputs a torsion free crystallographic group G , also known as a Bieberbach group. The second version also takes a vector v .

ResolutionCoxeterGroup(D, n) Inputs a Coxeter diagram D and an integer $n > 1$. It returns k terms of a free ZG -resolution for the Coxeter group of type D .

ResolutionDirectProduct(R, S) ResolutionDirectProduct($R, S, \text{"internal"}$) Inputs a ZG -resolution R and a ZH -resolution S . The second version also takes a boolean `internal`.

ResolutionExtension(g, R, S) ResolutionExtension($g, R, S, \text{"TestFiniteness"}$) ResolutionExtension($g, R, S, \text{"TestFiniteness", true}$) Inputs a group G , a ZG -resolution R , and a ZH -resolution S . The second version also takes a boolean `TestFiniteness`. The third version also takes a boolean `true`.

ResolutionFiniteDirectProduct(R, S) ResolutionFiniteDirectProduct($R, S, \text{"internal"}$) Inputs a ZG -resolution R and a ZH -resolution S . The second version also takes a boolean `internal`.

ResolutionFiniteExtension($\text{gensE}, \text{gensG}, R, n$) ResolutionFiniteExtension($\text{gensE}, \text{gensG}, R, n, \text{true}$) Inputs a list gensE of generators of a group E , a list gensG of generators of a group G , a ZG -resolution R , and an integer n . The second version also takes a boolean `true`.

ResolutionFiniteGroup(gens, n) ResolutionFiniteGroup($\text{gens}, n, \text{true}$) ResolutionFiniteGroup($\text{gens}, n, \text{true}, \text{true}$) Inputs a list gens of generators of a group G and an integer n . The second version also takes a boolean `true`. The third version also takes a boolean `true`.

ResolutionFiniteSubgroup(R, K) ResolutionFiniteSubgroup($R, \text{gensG}, \text{gensK}$) Inputs a ZG -resolution R for a group G and a subgroup K of finite index. The second version also takes a list gensG of generators of G and a list gensK of generators of K .

ResolutionGraphOfGroups(D, n) ResolutionGraphOfGroups(D, n, L) Inputs a graph of groups D and a positive integer n . The second version also takes a list L of subgroups.

ResolutionNilpotentGroup(G, n) ResolutionNilpotentGroup($G, n, \text{"TestFiniteness"}$) Inputs a nilpotent group G and an integer n . The second version also takes a string `TestFiniteness`.

ResolutionNormalSeries(L, n) ResolutionNormalSeries(L, n, true) ResolutionNormalSeries($L, n, \text{true}, \text{true}$) Inputs a list L of subgroups of a group G and an integer n . The second version also takes a boolean `true`. The third version also takes a boolean `true`.

ResolutionPrimePowerGroup(P, n) ResolutionPrimePowerGroup(G, n, p) Inputs a p -group P and an integer n . The second version also takes a group G and a prime p .

ResolutionSmallFpGroup(G, n) ResolutionSmallFpGroup(G, n, p) Inputs a small finitely presented group G and an integer n . The second version also takes a prime p .

ResolutionSubgroup(R, K) Inputs a ZG -resolution for an (infinite) group G and a subgroup K of finite index $|G : K|$. It returns a resolution of $\mathbb{Z}K$.

ResolutionSubnormalSeries(L, n) Inputs a positive integer n and a list $L = [L_1, \dots, L_k]$ of subgroups L_i of a finitely presented group G such that L_i is normal in L_{i+1} for $i = 1, \dots, k-1$. It returns a resolution of $\mathbb{Z}L_k$.

TwistedTensorProduct($R, S, \text{EhomG}, \text{GmapE}, \text{NhomE}, \text{NEhomN}, \text{EltsE}, \text{Mult}, \text{InvE}$) Inputs a ZG -resolution R , a ZH -resolution S , and a list of maps $\text{EhomG}, \text{GmapE}, \text{NhomE}, \text{NEhomN}, \text{EltsE}, \text{Mult}, \text{InvE}$ between the two resolutions.

RecalculateIncidenceNumbers(R) Inputs a ZG -resolution R which arises as the cellular chain complex of a regular CW-complex. It recalculates the incidence numbers.

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 fr

Chapter 3

Induced equivariant chain maps

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

Chapter 4

Functors

ExtendScalars(R, G, EltsG) Inputs a ZH -resolution R , a group G containing H as a subgroup, and a list EltsG of elements of G . It returns the GH -resolution $R \otimes G$.

HomToIntegers(X) Inputs either a ZG -resolution $X = R$, or an equivariant chain map $X = (F : R \rightarrow S)$. It returns the chain complex $\text{Hom}(X, \mathbb{Z})$.

HomToIntegersModP(R) Inputs a ZG -resolution R and returns the cochain complex obtained by applying $\text{Hom}(R, \mathbb{Z}/p\mathbb{Z})$ to the resolution.

HomToIntegralModule(R, f) Inputs a ZG -resolution R and a group homomorphism $f : G \rightarrow GL_n(\mathbb{Z})$ to the group G . It returns the G -cocomplex obtained by applying $\text{Hom}(R, \mathbb{Z})$ to the resolution.

HomToGModule(R, A) Inputs a ZG -resolution R and an abelian G -outer group A . It returns the G -cocomplex obtained by applying $\text{Hom}(R, A)$ to the resolution.

InduceScalars(R, hom) Inputs a ZQ -resolution R and a surjective group homomorphism $\text{hom} : G \rightarrow Q$. It returns the Q -resolution $R \otimes G$.

LowerCentralSeriesLieAlgebra(G) LowerCentralSeriesLieAlgebra(f) Inputs a pcg group G . If each $g \in G$ has a power g^p in the center of G , then f is the map $f(g) = g^p$. It returns the Lie algebra of the lower central series of G .

TensorWithIntegers(X) Inputs either a ZG -resolution $X = R$, or an equivariant chain map $X = (F : R \rightarrow S)$. It returns the chain complex $X \otimes \mathbb{Z}$.

FilteredTensorWithIntegers(R) Inputs a ZG -resolution R for which "filteredDimension" lies in NamesOfComplexes. It returns the chain complex $R \otimes \mathbb{Z}$.

TensorWithTwistedIntegers(X, rho) Inputs either a ZG -resolution $X = R$, or an equivariant chain map $X = (F : R \rightarrow S)$. It returns the chain complex $X \otimes \mathbb{Z}$ with the G -action twisted by rho .

TensorWithIntegersModP(X, p) Inputs either a ZG -resolution $X = R$, or a characteristic 0 chain complex, or an equivariant chain map $X = (F : R \rightarrow S)$. It returns the chain complex $X \otimes \mathbb{Z}/p\mathbb{Z}$.

TensorWithTwistedIntegersModP(X, p, rho) Inputs either a ZG -resolution $X = R$, or an equivariant chain map $X = (F : R \rightarrow S)$. It returns the chain complex $X \otimes \mathbb{Z}/p\mathbb{Z}$ with the G -action twisted by rho .

TensorWithRationals(R) Inputs a ZG -resolution R and returns the chain complex obtained by tensoring with \mathbb{Q} .

Chapter 5

Chain complexes

`ChainComplex(T)` Inputs a pure cubical complex, or cubical complex, or simplicial complex T and returns the (often) chain complex $C(T)$.

`ChainComplexOfPair(T, S)` Inputs a pure cubical complex or cubical complex T and contractible subcomplex S . It returns the chain complex $C(T, S)$.

`ChevalleyEilenbergComplex(X, n)` Inputs either a Lie algebra $X = A$ (over the ring of integers Z or over a field K) and an integer n . It returns the Chevalley-Eilenberg complex $C_n(X)$.

`LeibnizComplex(X, n)` Inputs either a Lie or Leibniz algebra $X = A$ (over the ring of integers Z or over a field K) and an integer n . It returns the Leibniz complex $C_n(X)$.

`SuspendedChainComplex(C)` Inputs a chain complex C and returns the chain complex S defined by applying the suspension operator S to C .

`ReducedSuspendedChainComplex(C)` Inputs a chain complex C and returns the chain complex S defined by applying the reduced suspension operator S to C .

`CoreducedChainComplex(C)` `CoreducedChainComplex(C, 2)` Inputs a chain complex C and returns a quasi-isomorphic coreduced chain complex.

`LefschetzNumber(F)` Inputs a chain map $F: C \rightarrow C$ with common source and target. It returns the Lefschetz number $L(F)$.

Chapter 6

Homology and cohomology groups

`Cohomology(X, n)` Inputs either a cochain complex $X = C$ (or G -cocomplex C) or a cochain map $X = (C \longrightarrow D)$ in
`CohomologyModule(C, n)` Inputs a G -cocomplex C together with a non-negative integer n . It returns the cohomology
`CohomologyPrimePart(C, n, p)` Inputs a cochain complex C in characteristic 0, a positive integer n , and a prime p .
`GroupCohomology(X, n)` `GroupCohomology(X, n, p)` Inputs a positive integer n and either a finite group $X = G$ or
`GroupHomology(X, n)`
`GroupHomology(X, n, p)` Inputs a positive integer n and either a finite group $X = G$ or a nilpotent Pcp-group $X = G$ or
`PersistentHomologyOfQuotientGroupSeries(S, n)`
`PersistentHomologyOfQuotientGroupSeries(S, n, p, ResolutionAlgorithm)` Inputs a positive integer n and
`PersistentCohomologyOfQuotientGroupSeries(S, n)`
`PersistentCohomologyOfQuotientGroupSeries(S, n, p, ResolutionAlgorithm)` Inputs a positive integer n and
`PersistentHomologyOfSubGroupSeries(S, n)`
`PersistentHomologyOfSubGroupSeries(S, n, p, ResolutionAlgorithm)` Inputs a positive integer n and a decre
`PersistentHomologyOfFilteredChainComplex(C, n, p)` Inputs a filtered chain complex C (of characteristic 0) and
`PersistentHomologyOfCommutativeDiagramOfPGroups(D, n)` Inputs a commutative diagram D of finite p -gro
`PersistentHomologyOfPureCubicalComplex(L, n, p)` Inputs a positive integer n , a prime p and an increasing cha
`ZZPersistentHomologyOfPureCubicalComplex(L, n, p)` Inputs a positive integer n , a prime p and any sequence L
`RipsHomology(G, n)` `RipsHomology(G, n, p)` Inputs a graph G , a non-negative integer n (and optionally a prime nu
`Barcode(P)` Inputs an integer persistence matrix P and returns the same information in the form of a binary matrix
`BarcodeDisplay(P)` `BarcodeDisplay(P, "mozilla")` Inputs an integer persistence matrix P , and an optional strin
`Homology(X, n)` Inputs either a chain complex $X = C$ or a chain map $X = (C \longrightarrow D)$. If $X = C$ then the torsion coeff
`HomologyPb(C, n)` This is a back-up function which might work in some instances where $Homology(C, n)$ fails. It is
`HomologyVectorSpace(X, n)` Inputs either a chain complex $X = C$ or a chain map $X = (C \longrightarrow D)$ in prime characte
`HomologyPrimePart(C, n, p)` Inputs a chain complex C in characteristic 0, a positive integer n , and a prime p . It ret
`LeibnizAlgebraHomology(A, n)` Inputs a Lie or Leibniz algebra $X = A$ (over the ring of integers \mathbb{Z} or over a field K) and
`LieAlgebraHomology(A, n)` Inputs a Lie algebra A (over the integers or a field) and a positive integer n . It returns th
`PrimePartDerivedFunctor(G, R, F, n)` Inputs a finite group G , a positive integer n , at least $n + 1$ terms of a ZP -res
`RankHomologyPGroup(G, n)` `RankHomologyPGroup(R, n)` `RankHomologyPGroup(G, n, "empirical")` Inputs a
`RankPrimeHomology(G, n)` Inputs a (smallish) p -group G together with a positive integer n . It returns a function dim

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 $p(x)$ and $q(x)$ are polynomials.
`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 $f(x) = P(x)/Q(x)$ whose denominator $Q(x)$ is a product of cyclotomic polynomials.
`PoincareSeriesPrimePart(G, p, n)` Inputs a finite group G , a prime p , and a positive integer n . It returns a quotient of polynomials $f(x) = P(x)/Q(x)$ whose denominator $Q(x)$ is a product of cyclotomic polynomials.
`PoincareSeriesLHS(G)` Inputs a finite 2-group G and returns a quotient of polynomials $f(x) = P(x)/Q(x)$ whose denominator $Q(x)$ is a product of cyclotomic 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 available)
`IntegralRingGenerators (R, n)` Inputs at least $n + 1$ terms of a ZG -resolution and integer $n > 0$. It returns a minimal set of generators.
`ModPCohomologyGenerators (G, n)`
`ModPCohomologyGenerators (R)` Inputs either a p -group G and positive integer n , or else n terms of a minimal ZG -resolution.
`ModPCohomologyRing (G, n)`
`ModPCohomologyRing (G, n, level)`
`ModPCohomologyRing (R)`
`ModPCohomologyRing (R, level)` Inputs either a p -group G and positive integer n , or else n terms of a minimal ZG -resolution.
`ModPRingGenerators (A)` Inputs a mod p cohomology ring A (created using the preceding function). It returns a minimal set of generators.
`Mod2CohomologyRingPresentation (G)`
`Mod2CohomologyRingPresentation (G, n)`
`Mod2CohomologyRingPresentation (A)`
`Mod2CohomologyRingPresentation (R)` When applied to a finite 2-group G this function returns a presentation of the mod 2 cohomology ring.

Chapter 9

Cohomology rings of p -groups (mainly $p = 2$)

The functions on this page were written by PAUL SMITH. (They are included in HAP but they are also independently included in Paul Smiths HAPprime package.)

•

<code>Mod2CohomologyRingPresentation(G)</code>	
<code>Mod2CohomologyRingPresentation(G,n)</code>	
<code>Mod2CohomologyRingPresentation(A)</code>	
<code>Mod2CohomologyRingPresentation(R)</code>	When applied to a finite 2-group G this function returns a presentation
<code>PoincareSeriesLHS(G)</code>	Inputs a finite 2-group G and returns a quotient of polynomials $f(x) = P(x)/Q(x)$ whose

Chapter 10

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 $M^{(c)}(G) = \gamma_c(G) / \gamma_{c+1}(G)$.

`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)` Inputs a finite group G and normal subgroup N and returns a subgroup $Z^*(G, N)$ of the centre of N .

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

- `NonabelianSymmetricKernel(G)`
- `NonabelianSymmetricKernel(G, m)` Inputs a finite or nilpotent infinite group G and returns the abelian invariant $\gamma_m(G)$.
- `NonabelianSymmetricSquare(G)`
- `NonabelianSymmetricSquare(G, m)` Inputs a finite or nilpotent infinite group G and returns a record T with the following fields:

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

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

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

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

`ThirdHomotopyGroupOfSuspensionB(G)`

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

`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 11

Lie commutators and nonabelian Lie tensors

•
Functions on this page are joint work with HAMID MOHAMMADZADEH, and implemented by him.

`LieCoveringHomomorphism(L)` Inputs a finite dimensional Lie algebra L over a field, and returns a surjective Lie

`LeibnizQuasiCoveringHomomorphism(L)` Inputs a finite dimensional Lie algebra L over a field, and returns a sur

`LieEpiCentre(L)` Inputs a finite dimensional Lie algebra L over a field, and returns an ideal $Z^*(L)$ of the centre of

`LieExteriorSquare(L)` Inputs a finite dimensional Lie algebra L over a field. It returns a record E with the follow

`LieTensorSquare(L)` Inputs a finite dimensional Lie algebra L over a field and returns a record T with the follow

`LieTensorCentre(L)` Inputs a finite dimensional Lie algebra L over a field and returns the largest ideal N such th

Chapter 12

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 co

`IdentityAmongRelatorsDisplay(R,n)` `IdentityAmongRelatorsDisplay(R,n,"mozilla")` Inputs a free

`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

`PresentationOfResolution(R)` Inputs at least two terms of a reduced ZG -resolution R and returns a record P w

`TorsionGeneratorsAbelianGroup(G)` Inputs an abelian group G and returns a generating set $[x_1, \dots, x_n]$ where

Orbit polytopes and fundamental domains

VectorStabilizer(G, v) Inputs a permutation group or matrix group G of degree n and a rational vector of degree n .

Chapter 14

Cocycles

•

`CcGroup(A, f)` Inputs a G -module A (i.e. an abelian G -outer group) and a standard 2-cocycle $f: G \times G \rightarrow A$. It returns an integer matrix M with the following property: $M^{-1}fM$ is a standard 2-cocycle.

`CocycleCondition(R, n)` Inputs a resolution R and an integer $n > 0$. It returns an integer matrix M with the following property: $M^{-1}fM$ is a standard 2-cocycle.

`StandardCocycle(R, f, n)` Inputs a ZG -resolution R (with contracting homotopy), a positive integer n and an integer matrix f . It returns an integer matrix M with the following property: $M^{-1}fM$ is a standard 2-cocycle.

`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 an integer matrix M with the following property: $M^{-1}fM$ is a standard 2-cocycle.

Chapter 15

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 , it returns $v + w$ 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 $v + w$ modulo p .

`AlgebraicReduction(w)` Inputs a word w in a free ZG -module and returns a reduced version of the word in w .

`AlgebraicReduction(w, p)` Inputs a word w in a free ZG -module and returns a reduced version of the word in w 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 modulo S .

`ResolutionBoundaryOfWord(R, n, w)` Inputs a resolution R , a positive integer n and a list w representing a word in the free ZG -module. It returns the boundary of the word w in the resolution R .

Chapter 16

FpG-modules

`CompositionSeriesOfFpGModules(M)` Inputs an *FpG*-module M and returns a list of *FpG*-modules that constitute a composition series for M .

`DirectSumOfFpGModules(M, N)`

`DirectSumOfFpGModules([M[1], M[2], ..., M[k]])` Inputs two *FpG*-modules M and N with common group G and returns their direct sum.

`FpGModule(A, P)`

`FpGModule(A, G, p)` Inputs a p -group P and a matrix A whose rows have length a multiple of the order of G . It returns the *FpG*-module defined by A over P .

`FpGModuleDualBasis(M)` Inputs an *FpG*-module M . It returns a record R with two components: $R.freeModule$ is a free F -module isomorphic to M and $R.dualBasis$ is a dual basis for M .

`FpGModuleHomomorphism(M, N, A)`

`FpGModuleHomomorphismNC(M, N, A)` Inputs *FpG*-modules M and N over a common p -group G . Also inputs a list A of matrices representing the homomorphism.

`DesuspensionFpGModule(M, n)`

`DesuspensionFpGModule(R, 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.

`RadicalSeriesOfFpGModule(M)` Inputs an *FpG*-module M and returns a list of *FpG*-modules that constitute the radical series of M .

`GeneratorsOfFpGModule(M)` Inputs an *FpG*-module M and returns a matrix whose rows correspond to a minimal generating set for M .

`ImageOfFpGModuleHomomorphism(f)` Inputs an *FpG*-module homomorphism $f : M \rightarrow N$ and returns its image.

`GroupAlgebraAsFpGModule(G)` Inputs a p -group G and returns its mod p group algebra as an *FpG*-module.

`IntersectionOfFpGModules(M, N)` Inputs two *FpG*-modules M, N arising as submodules in a common free module and returns their intersection.

`IsFpGModuleHomomorphismData(M, N, A)` Inputs *FpG*-modules M and N over a common p -group G . Also inputs a list A of matrices representing the homomorphism. Returns a record with fields `isHomomorphism` and `data`.

`MaximalSubmoduleOfFpGModule(M)` Inputs an *FpG*-module M and returns one maximal *FpG*-submodule of M .

`MaximalSubmodulesOfFpGModule(M)` Inputs an *FpG*-module M and returns the list of maximal *FpG*-submodules of M .

`MultipleOfFpGModule(w, M)` Inputs an *FpG*-module M and a list $w := [g_1, \dots, g_t]$ of elements in the group $G = M$. It returns the multiple of M by w .

`ProjectedFpGModule(M, k)` Inputs an *FpG*-module M of ambient dimension $n|G|$, and an integer k between 1 and n . It returns the projection of M onto a subspace of dimension $k|G|$.

`RandomHomomorphismOfFpGModules(M, N)` Inputs two *FpG*-modules M and N over a common group G . It returns a random homomorphism from M to N .

`Rank(f)` Inputs an *FpG*-module homomorphism $f : M \rightarrow N$ and returns the dimension of the image of f as a vector space over F .

`SumOfFpGModules(M, N)` Inputs two *FpG*-modules M, N arising as submodules in a common free module $(FG)^n$ and returns their sum.

`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 the words in M corresponding to the vectors in L .

Chapter 17

Meataxe modules

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

Chapter 18

G-Outer Groups

`GOuterGroup(E, N)`
`GOuterGroup()` **Inputs** a group E and normal subgroup N . It returns N as a G -outer group where $G = E/N$. The fun
`GOuterGroupHomomorphismNC(A, B, phi)`
`GOuterGroupHomomorphismNC()` **Inputs** G -outer groups A and B with common acting group, and a group homomor
`GOuterHomomorphismTester(A, B, phi)` **Inputs** G -outer groups A and B with common acting group, and a group ho
`Centre(A)` **Inputs** G -outer group A and returns the group theoretic centre of `ActedGroup(A)` as a G -outer group.
`DirectProductGog(A, B)`
`DirectProductGog(Lst)` **Inputs** G -outer groups A and B with common acting group, and returns their group-theore

Chapter 19

Cat-1-groups

`AutomorphismGroupAsCatOneGroup(G)` Inputs a group G and returns the Cat-1-group C corresponding to the crossed module $(G, \text{Aut}(G))$.

`HomotopyGroup(C, n)` Inputs a cat-1-group C and an integer n . It returns the n th homotopy group of C .

`HomotopyModule(C, 2)` Inputs a cat-1-group C and an integer $n=2$. It returns the second homotopy group of C as a C -module.

`ModuleAsCatOneGroup(G, alpha, M)` Inputs a group G , an abelian group M and a homomorphism $\alpha: G \rightarrow \text{Aut}(M)$. It returns the Cat-1-group C corresponding to the crossed module (M, α) .

`MooreComplex(C)` Inputs a cat-1-group C and returns its Moore complex $[M_1 \rightarrow M_0]$ as a list whose single entry is the Moore complex.

`NormalSubgroupAsCatOneGroup(G, N)` Inputs a group G with normal subgroup N . It returns the Cat-1-group C corresponding to the crossed module $(N, \text{Inn}(G))$.

Chapter 20

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 web browser.

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

`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 group is spherical.

`CoxeterDiagramMatrix(D)` Inputs a Coxeter diagram D and returns a matrix representation of it. The matrix is the Coxeter matrix.

`CoxeterSubDiagram(D, V)` Inputs a Coxeter diagram D and a subset V of its vertices. It returns the full sub-diagram induced by 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 and xy has even length.

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

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

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

Chapter 21

Simplicial Complexes

`Homology(T,n)` `Homology(T)` Inputs a pure cubical complex, or cubical complex, or simplicial complex T and a non-negative integer n . Returns the n -th homology group of T .

`RipsHomology(G,n)` `RipsHomology(G,n,p)` Inputs a graph G , a non-negative integer n (and optionally a prime number p). Returns the n -th Rips homology group of G .

`Bettinnumbers(T,n)` `Bettinnumbers(T)` Inputs a pure cubical complex, or cubical complex, simplicial complex or chain complex T and a non-negative integer n . Returns the n -th Bettin number of T .

`ChainComplex(T)` Inputs a pure cubical complex, or cubical complex, or simplicial complex T and returns the (often infinite) chain complex of T .

`CechComplexOfPureCubicalComplex(T)` Inputs a d -dimensional pure cubical complex T and returns a simplicial complex.

`RipsChainComplex(S,epsilon)` `RipsChainComplex(S,epsilon,true)` Inputs an $n \times n$ symmetric matrix S with real entries and a non-negative real number ϵ . Returns the Rips chain complex of S with respect to ϵ .

`VectorsToSymmetricMatrix(M)` `VectorsToSymmetricMatrix(M,distance)` Inputs a matrix M of rational numbers and a non-negative real number ϵ . Returns the symmetric matrix S such that $S_{ij} = 1$ if $|M_{ij} - M_{ji}| \leq \epsilon$ and 0 otherwise.

`EulerCharacteristic(T)` Inputs a pure cubical complex, or cubical complex, or simplicial complex T and returns the Euler characteristic of T .

`MaximalSimplicesToSimplicialComplex(L)` Inputs a list L whose entries are lists of vertices representing the maximal simplices of a simplicial complex. Returns the simplicial complex.

`SkeletonOfSimplicialComplex(S,k)` Inputs a simplicial complex S and a positive integer k less than or equal to the dimension of S . Returns the k -skeleton of S .

`GraphOfSimplicialComplex(S)` Inputs a simplicial complex S and returns the graph of S .

`ContractibleSubcomplexOfSimplicialComplex(S)` Inputs a simplicial complex S and returns a (probably maximal) contractible subcomplex of S .

`PathComponentsOfSimplicialComplex(S,n)` Inputs a simplicial complex S and a nonnegative integer n . If $n = 0$ returns the number of path components of S . If $n > 0$ returns the number of path components of the n -skeleton of S .

`QuillenComplex(G)` Inputs a finite group G and returns, as a simplicial complex, the order complex of the poset of proper subgroups of G .

`SymmetricMatrixToIncidenceMatrix(S,t)` Inputs a symmetric integer matrix S and an integer t . It returns the incidence matrix of the t -skeleton of the simplicial complex defined by S .

`IncidenceMatrixToGraph(M)` Inputs a symmetric 0/1 matrix M . It returns the graph with one vertex for each row of M and edges between vertices i and j if $M_{ij} = 1$.

`PathComponentsOfGraph(G,n)` Inputs a graph G and a nonnegative integer n . If $n = 0$ the number of path components of G . If $n > 0$ returns the number of path components of the n -skeleton of G .

`ContractGraph(G)` Inputs a graph G and tries to remove vertices and edges to produce a smaller graph G' such that G' is homotopy equivalent to G .

`GraphDisplay(G)` This function uses `GraphViz` software to display a graph G .

`SimplicialMap(K,L,f)` `SimplicialMapNC(K,L,f)` Inputs simplicial complexes K, L and a function $f: K \rightarrow L$. Returns a simplicial map from K to L .

`ChainMapOfSimplicialMap(f)` Inputs a simplicial map $f: K \rightarrow L$ and returns the corresponding chain map $C_*(f): C_*(K) \rightarrow C_*(L)$.

`SimplicialNerveOfGraph(G,d)` Inputs a graph G and returns a d -dimensional simplicial complex K whose 1-skeleton is G .

Chapter 22

Cubical Complexes

`ArrayToPureCubicalComplex(A,n)` Inputs an integer array A of dimension d and an integer n . It returns a d -dimensional pure cubical complex.

`PureCubicalComplex(A,n)` Inputs a binary array A of dimension d . It returns the corresponding d -dimensional pure cubical complex.

`PureCubicalComplexIntersection(S,T)` Inputs two pure cubical complexes with common dimension and array size. It returns their intersection.

`PureCubicalComplexUnion(S,T)` Inputs two pure cubical complexes with common dimension and array size. It returns their union.

`PureCubicalComplexDifference(S,T)` Inputs two pure cubical complexes with common dimension and array size. It returns their difference.

`ReadImageAsPureCubicalComplex("file.png",n)` Reads an image file ("file.png", "file.eps", "file.bmp" etc) and returns a pure cubical complex.

`ReadImageSequenceAsPureCubicalComplex("directory",n)` Reads the name of a directory containing a sequence of image files and returns a pure cubical complex.

`Size(T)` This returns the number of non-zero entries in the binary array of the pure cubical complex T .

`WritePureCubicalComplexAsImage(T,"filename","ext")` Inputs a 2-dimensional pure cubical complex T , and writes it to a file.

`ViewPureCubicalComplex(T)` `ViewPureCubicalComplex(T,"mozilla")` Inputs a 2-dimensional pure cubical complex T , and displays it in a window.

`Homology(T,n)` `Homology(T)` Inputs a pure cubical complex, or cubical complex, or simplicial complex T and an integer n . It returns the n -th homology group.

`Bettinnumbers(T,n)` `Bettinnumbers(T)` Inputs a pure cubical complex, or cubical complex, simplicial complex T and an integer n . It returns the n -th Bettin number.

`DirectProductOfPureCubicalComplexes(M,N)` Inputs two cubical complexes M, N and returns their direct product.

`EulerCharacteristic(T)` Inputs a pure cubical complex, or cubical complex, or simplicial complex T and returns its Euler characteristic.

`PathComponentOfPureCubicalComplex(T,n)` Inputs a pure cubical complex T and an integer n in the range 1, ..., $\dim T$. It returns the n -th path component.

`ChainComplex(T)` Inputs a pure cubical complex, or cubical complex, or simplicial complex T and returns the (often infinite) chain complex.

`ChainComplexOfPair(T,S)` Inputs a pure cubical complex or cubical complex T and subcomplex S . It returns the chain complex of the pair.

`ExcisedPureCubicalPair(T,S)` Inputs a pure cubical complex T and subcomplex S . It returns the pair $[T \setminus \text{int} S, S]$.

`ChainInclusionOfPureCubicalPair(S,T)` Inputs a pure cubical complex T and subcomplex S . It returns the chain map of the inclusion.

`ChainMapOfPureCubicalPairs(M,S,N,T)` Inputs a pure cubical complex N and subcomplexes M, T and S in T . It returns the chain map.

`ContractPureCubicalComplex(T)` Inputs a pure cubical complex T of dimension d and removes d -dimensional cells.

`ContractedComplex(T)` Inputs a pure cubical complex T and returns a structural copy of the complex obtained from T by contracting all d -cells.

`ContractibleSubcomplexOfPureCubicalComplex(T)` Inputs a pure cubical complex T and returns a maximal contractible subcomplex.

`AcyclicSubcomplexOfPureCubicalComplex(T)` Inputs a pure cubical complex T and returns a (not necessarily contractible) acyclic subcomplex.

`HomotopyEquivalentMaximalPureCubicalSubcomplex(T,S)` Inputs a pure cubical complex T together with a subcomplex S . It returns a maximal subcomplex homotopy equivalent to S .

`HomotopyEquivalentMinimalPureCubicalSubcomplex(T,S)` Inputs a pure cubical complex T together with a subcomplex S . It returns a minimal subcomplex homotopy equivalent to S .

`BoundaryOfPureCubicalComplex(T)` Inputs a pure cubical complex T and returns its boundary as a pure cubical complex.

`SingularitiesOfPureCubicalComplex(T,radius,tolerance)` Inputs a pure cubical complex T together with a radius and tolerance. It returns the set of singularities.

`ThickenedPureCubicalComplex(T)` Inputs a pure cubical complex T and returns a pure cubical complex S . If a cell σ of T is k -dimensional, then σ is a cell of S of dimension $k+1$.

`MorseFiltration(M,i,t,bool)` `MorseFiltration(M,i,t)` Inputs a pure cubical complex M of dimension d , an integer i , a real number t , and a boolean. It returns the i -th Morse filtration.

`ComplementOfPureCubicalComplex(T)` Inputs a pure cubical complex T and returns a pure cubical complex S . $A \cap S = \emptyset$ and $A \cup S$ is the smallest cubical complex containing A and S .

`PureCubicalComplexToTextFile(file,M)` Inputs a pure cubical complex M and a string containing the address of a file. It writes the complex to the file.

Chapter 23

Commutative diagrams and abstract categories

COMMUTATIVE DIAGRAMS

`HomomorphismChainToCommutativeDiagram(H)` Inputs a list $H = [h_1, h_2, \dots, h_n]$ of mappings such that the composition of h_i is the identity.
`NormalSeriesToQuotientDiagram(L)` `NormalSeriesToQuotientDiagram(L, M)` Inputs an increasing (or decreasing) normal series L of a group G and a normal subgroup M of G .
`NerveOfCommutativeDiagram(D)` Inputs a commutative diagram D and returns the commutative diagram ND corresponding to D .
`GroupHomologyOfCommutativeDiagram(D, n)` `GroupHomologyOfCommutativeDiagram(D, n, prime)` `GroupHomologyOfCommutativeDiagramOfPGroups(D, n)` Inputs a commutative diagram D of finite p -groups and returns the n -th homology group of D .

ABSTRACT CATEGORIES

`CategoricalEnrichment(X, Name)` Inputs a structure X such as a group or group homomorphism, together with a name $Name$ for the category, and returns the categorical enrichment of X .
`IdentityArrow(X)` Inputs an object X in some category, and returns the identity arrow on the object X .
`InitialArrow(X)` Inputs an object X in some category, and returns the arrow from the initial object in the category to X .
`TerminalArrow(X)` Inputs an object X in some category, and returns the arrow from X to the terminal object in the category.
`HasInitialObject(Name)` Inputs the name of a category and returns true or false depending on whether the category has an initial object.
`HasTerminalObject(Name)` Inputs the name of a category and returns true or false depending on whether the category has a terminal object.
`Source(f)` Inputs an arrow f in some category, and returns its source.
`Target(f)` Inputs an arrow f in some category, and returns its target.
`CategoryName(X)` Inputs an object or arrow X in some category, and returns the name of the category.
`"*", "=", "+", "-"` Composition of suitable arrows f, g is given by $f * g$ when the source of f equals the target of g .
`Object(X)` Inputs an object X in some category, and returns the GAP structure Y such that $X = \text{CategoricalEnrichment}(Y)$.
`Mapping(X)` Inputs an arrow f in some category, and returns the GAP structure Y such that $f = \text{CategoricalEnrichment}(Y)$.
`IsCategoryObject(X)` Inputs X and returns true if X is an object in some category.
`IsCategoryArrow(X)` Inputs X and returns true if X is an arrow in some category.

Chapter 24

Arrays and Pseudo lists

`Array(A, f)` Inputs an array A and a function f . It returns the array obtained by applying f to each entry of A .

`ArrayDimension(A)` Inputs an array A and returns its dimension.

`ArrayDimensions(A)` Inputs an array A and returns its dimensions.

`ArraySum(A)` Inputs an array A and returns the sum of its entries.

`ArrayValue(A, x)` Inputs an array A and a coordinate vector x . It returns the value of the entry in A with coordinate x .

`ArrayValueFunctions(d)` Inputs a positive integer d and returns an efficient version of the function `ArrayValue` for dimension d .

`ArrayAssign(A, x, n)` Inputs an array A and a coordinate vector x and an integer n . It sets the entry of A with coordinate x to n .

`ArrayAssignFunctions(d)` Inputs a positive integer d and returns an efficient version of the function `ArrayAssign` for dimension d .

`ArrayIterate(d)` Inputs a positive integer d and returns a function `ArrayIt(Dimensions, f)`. This function inputs a coordinate vector x and returns $f(\text{ArrayValue}(A, x))$.

`BinaryArrayToTextFile(file, A)` Inputs a string containing the address of a file, and an array A of 0s and 1s. It writes the array A to the file.

`FrameArray(A)` Inputs an array A and returns the array obtained by appending a 0 to the beginning and end of each row.

`UnframeArray(A)` Inputs an array A and returns the array obtained by removing the first and last entry in each row.

`Add(L, x)` Let L be a pseudo list of length n , and x an object compatible with the entries in L . If x is not in L then this operation appends x to the end of L .

`Append(L, K)` Let L be a pseudo list and K a list whose objects are compatible with those in L . This operation appends the elements of K to the end of L .

`ListToPseudoList(L)` Inputs a list L and returns the pseudo list representation of L .

Chapter 25

Parallel Computation - Core Functions

```
ChildProcess()  
ChildProcess("computer.ac.wales")  
ChildProcess(["-m", "100000M", "-T"])  
ChildProcess("computer.ac.wales", ["-m", "100000M", "-T"]) This starts a GAP session as a child process
```

```
- open a shell on thishost  
- cd .ssh  
- ls  
-> if id_dsa, id_rsa etc exists, skip the next two steps!  
- ssh-keygen -t rsa  
- ssh-keygen -t dsa  
- scp *.pub user@remotehost:~/  
- ssh remotehost -l user  
- cat id_rsa.pub >> .ssh/authorized_keys  
- cat id_dsa.pub >> .ssh/authorized_keys  
- rm id_rsa.pub id_dsa.pub  
- exit
```

You should now be able to connect from "thishost" to "remotehost" without a password prompt.)

`ChildClose(s)` This closes the stream `s` to a child GAP process.

`ChildCommand("cmd;", s)` This runs a GAP command "cmd;" on the child process accessed by the stream `s`. Here

`NextAvailableChild(L)` Inputs a list `L` of child processes and returns a child in `L` which is ready for computation

`IsAvailableChild(s)` Inputs a child process `s` and returns true if `s` is currently available for computations, and false otherwise.

`ChildPut(A, "B", s)` This copies a GAP object `A` on the parent process to an object `B` on the child process `s`. (The

`ChildGet("A", s)` This function copies a GAP object `A` on the child process `s` and returns it on the parent process

`HAPPrintTo("file", R)` Inputs a name "file" of a new text file and a HAP object `R`. It writes the object `R` to "file".

`HAPRead("file", R)` Inputs a name "file" containing a HAP object `R` and returns the object. Currently this is only in

Chapter 26

Parallel Computation - Extra Functions

`ChildFunction("function(arg);", s)` This runs the GAP function "function(arg);" on a child process accessed by `s`.
`ChildRead(s)` This returns, as a string, the output of the last application of `ChildFunction("function(arg);", s)`.
`ChildReadEval(s)` This returns, as an evaluated string, the output of the last application of `ChildFunction("function(arg);", s)`.
`ParallelList(I, fn, L)` Inputs a list I , a function fn such that $fn(x)$ is defined for all x in I , and a list of children L .

Chapter 27

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)` Inputs a resolution, chain complex or cochain complex C and returns the function $C!.dimension$.
`Dimension(M)` Inputs a resolution, chain complex or cochain complex C and returns the function $C!.dimension$.
`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 object).
`Source(f)` Inputs a chain map, or cochain map, or equivariant chain map, or FpG -module homomorphism f and returns the source object of f .
`Target(f)` Inputs a chain map, or cochain map, or equivariant chain map, or FpG -module homomorphism f and returns the target object of f .

Chapter 28

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 1

`Classify(L, Inv)` Inputs a list of objects L and a function Inv which computes an invariant of each object. It returns

`RefineClassification(C, Inv)` Inputs a list $C := Classify(L, OldInv)$ and returns a refined classification according

`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 $f : G \longrightarrow GL(n, \mathbb{C})$

`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

`LinearHomomorphismsPersistenceMat(L)` Inputs a composable sequence L of vector space homomorphisms. It returns a persistence matrix

`NormalSeriesToQuotientHomomorphisms(L)` Inputs an (increasing or decreasing) chain L of normal subgroups of a group G

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

Index

- AcyclicSubomplexOfPureCubicalComplex, 24
- Add, 26
- AddFreeWords, 17
- AddFreeWordsModP, 17
- AlgebraicReduction, 17
- Append, 26
- Array, 26
- ArrayAssign, 26
- ArrayAssignFunctions, 26
- ArrayDimension, 26
- ArrayDimensions, 26
- ArrayIterate, 26
- ArraySum, 26
- ArrayToPureCubicalComplex, 24
- ArrayValue, 26
- ArrayValueFunctions, 26
- AutomorphismGroupAsCatOneGroup, 21

- BaerInvariant, 12
- BarCode, 8
- BarCodeDisplay, 8
- Bettinnumbers, 23, 24
- BigStepLCS, 30
- BinaryArrayToTextFile, 26
- BoundaryMap, 29
- BoundaryMatrix, 29
- BoundaryOfPureCubicalComplex, 24

- CategoricalEnrichment, 25
- CategoryName, 25
- CayleyGraphDisplay, 14
- CcGroup (HAPcocyclic), 16
- CechComplexOfPureCubicalComplex, 23
- Centre, 20
- ChainComplex, 7
- ChainComplexOfPair, 7
- ChainInclusionOfPureCubicalPair, 24
- ChainMapOfPureCubicalPairs, 24
- ChainMapOfSimplicialMap, 23

- ChevalleyEilenbergComplex, 7
- ChildClose, 27
- ChildCommand, 27
- ChildFunction, 28
- ChildGet, 27
- ChildProcess, 27
- ChildPut, 27
- ChildRead, 28
- ChildReadEval, 28
- Classify, 30
- Coclass, 12
- CocycleCondition, 16
- Cohomology, 8
- CohomologyModule, 8
- CohomologyPrimePart, 8
- ComplementOfPureCubicalComplex, 24
- Compose(f,g), 30
- CompositionSeriesOfFpGModules, 18
- ContractedComplex, 24
- ContractGraph, 23
- ContractibleGcomplex, 15
- ContractibleSubcomplexOfSimplicialComplex, 23
- ContractibleSubomplexOfPureCubicalComplex, 24
- ContractPureCubicalComplex, 24
- CoreducedChainComplex, 7
- CoxeterComplex, 15
- CoxeterDiagramComponents, 22
- CoxeterDiagramDegree, 22
- CoxeterDiagramDisplay, 22
- CoxeterDiagramFpArtinGroup, 22
- CoxeterDiagramFpCoxeterGroup, 22
- CoxeterDiagramIsSpherical, 22
- CoxeterDiagramMatrix, 22
- CoxeterDiagramVertices, 22
- CoxeterSubDiagram, 22

- DesuspensionFpGModule, 18

- DesuspensionMtxModule, 19
- Dimension, 29
- DirectProductGog, 20
- DirectProductOfPureCubicalComplexes, 24
- DirectSumOfFpGModules, 18
- EpiCentre, 12
- EquivariantChainMap, 5
- EulerCharacteristic, 23
- EvaluateProperty, 29
- EvenSubgroup, 22
- ExpansionOfRationalFunction, 9
- ExtendScalars, 6
- FilteredTensorWithIntegers, 6
- FpGModule, 18
- FpGModuleDualBasis, 18
- FpGModuleHomomorphism, 18
- FpG_to_MtxModule, 19
- FrameArray, 26
- FreeGResolution, 3
- FundamentalDomainStandardSpaceGroup
(HAPcryst), 15
- GeneratorsOfFpGModule, 18
- GeneratorsOfMtxModule, 19
- GOuterGroup, 20
- GOuterGroupHomomorphismNC, 20
- GOuterHomomorphismTester, 20
- GraphDisplay, 23
- GraphOfGroupsDisplay, 22
- GraphOfGroupsTest, 22
- GraphOfSimplicialComplex, 23
- GroupAlgebraAsFpGModule, 18
- GroupCohomology, 8
- GroupHomology, 8
- GroupHomologyOfCommutativeDiagram, 25
- GroupOfResolution, 29
- HAPcopyright, 30
- HAPPrintTo, 27
- HAPRead, 27
- HasInitialObject, 25
- HasTerminalObject, 25
- Homology, 8, 24
- HomologyPb, 8
- HomologyPrimePart, 8
- HomologyVectorSpace, 8
- HomomorphismChainToCommutativeDiagram, 25
- HomotopyEquivalentMaximalPureCubicalSubcomplex, 24
- HomotopyEquivalentMinimalPureCubicalSubcomplex, 24
- HomotopyGroup, 21
- HomotopyModule, 21
- HomToGModule, 6
- HomToIntegers, 6
- HomToIntegersModP, 6
- HomToIntegralModule, 6
- IdentityAmongRelatorsDisplay, 14
- IdentityArrow, 25
- ImageOfFpGModuleHomomorphism, 18
- IncidenceMatrixToGraph, 23
- InduceScalars, 6
- InitialArrow, 25
- IntegralCupProduct, 10
- IntegralRingGenerators, 10
- IntersectionOfFpGModules, 18
- IsAspherical, 14
- IsAvailableChild, 27
- IsCategoryArrow, 25
- IsCategoryObject, 25
- IsFpGModuleHomomorphismData, 18
- IsLieAlgebraHomomorphism, 30
- IsSuperperfect, 30
- LefschetzNumber, 7
- LeibnizAlgebraHomology, 8
- LeibnizComplex, 7
- LeibnizQuasiCoveringHomomorphism, 13
- Length, 29
- LieAlgebraHomology, 8
- LieCoveringHomomorphism, 13
- LieEpiCentre, 13
- LieExteriorSquare, 13
- LieTensorCentre, 13
- LieTensorSquare, 13
- LinearHomomorphismsPersistenceMat, 30
- ListToPseudoList, 26
- LowerCentralSeriesLieAlgebra, 6
- MakeHAPManual, 30
- Map, 29
- Mapping, 25

- MaximalSimplicesToSimplicialComplex, 23
- MaximalSubmoduleOfFpGModule, 18
- MaximalSubmodulesOfFpGModule, 18
- Mod2CohomologyRingPresentation (HAP-prime), 11
- ModPCohomologyGenerators, 10
- ModPCohomologyRing, 10
- ModPRingGenerators, 10
- ModuleAsCatOneGroup, 21
- MooreComplex, 21
- MorseFiltration, 24
- MultipleOfFpGModule, 18
- MultiplyWord, 17
- Negate, 17
- NegateWord, 17
- NerveOfCommutativeDiagram, 25
- NextAvailableChild, 27
- NonabelianExteriorProduct, 12
- NonabelianSymmetricKernel, 12
- NonabelianSymmetricSquare, 12
- NonabelianTensorProduct, 12
- NonabelianTensorSquare, 12
- NormalSeriesToQuotientDiagram, 25
- NormalSeriesToQuotientHomomorphisms, 30
- NormalSubgroupAsCatOneGroup, 21
- Object, 25
- OrbitPolytope, 15
- ParallelList, 28
- PathComponentOfPureCubicalComplex, 24
- PathComponentsOfGraph, 23
- PathComponentsOfSimplicialComplex, 23
- PermToMatrixGroup, 30
- PersistentCohomologyOfQuotientGroupSeries, 8
- PersistentHomologyOfCommutativeDiagramOfPGroups, 25
- PersistentHomologyOfFilteredChainComplex, 8
- PersistentHomologyOfPureCubicalComplex, 8
- PersistentHomologyOfQuotientGroupSeries, 8
- PersistentHomologyOfSubGroupSeries, 8
- PoincareSeries, 9
- PoincareSeriesLHS (HAPprime), 11
- PoincareSeriesPrimePart, 9
- PolytopalComplex, 15
- PolytopalGenerators, 15
- Prank, 9
- PresentationOfResolution, 14
- PrimePartDerivedFunctor, 8
- PrintZGword, 17
- ProjectedFpGModule, 18
- PureCubicalComplex, 24
- PureCubicalComplexDifference, 24
- PureCubicalComplexIntersection, 24
- PureCubicalComplexToTextFile, 24
- PureCubicalComplexUnion, 24
- QuillenComplex, 23
- RadicalOfFpGModule, 18
- RadicalSeriesOfFpGModule, 18
- RandomHomomorphismOfFpGModules, 18
- Rank, 18
- RankHomologyPGroup, 8
- RankPrimeHomology, 8
- ReadImageAsPureCubicalComplex, 24
- ReadImageSequenceAsPureCubicalComplex, 24
- RecalculateIncidenceNumbers, 3
- ReducedSuspendedChainComplex, 7
- RefineClassification, 30
- RelativeSchurMultiplier, 12
- ResolutionAbelianGroup, 3
- ResolutionAlmostCrystalGroup, 3
- ResolutionAlmostCrystalQuotient, 3
- ResolutionArtinGroup, 3
- ResolutionAsphericalPresentation, 3
- ResolutionBieberbachGroup (HAPcryst), 3
- ResolutionBoundaryOfWord, 17
- ResolutionCoxeterGroup, 3
- ResolutionDirectProduct, 3
- ResolutionExtension, 3
- ResolutionFiniteDirectProduct, 3
- ResolutionFiniteExtension, 3
- ResolutionFiniteGroup, 3
- ResolutionFiniteSubgroup, 3
- ResolutionFpGModule, 4
- ResolutionGraphOfGroups, 3
- ResolutionNilpotentGroup, 3
- ResolutionNormalSeries, 3
- ResolutionPrimePowerGroup, 3
- ResolutionSmallFpGroup, 3
- ResolutionSubgroup, 3
- ResolutionSubnormalSeries, 3
- RipsChainComplex, 23

[RipsHomology](#), 8
[SimplicialMap](#), 23
[SimplicialMapNC](#), 23
[SimplicialNerveOfGraph](#), 23
[SingularitiesOfPureCubicalComplex](#), 24
[SkeletonOfSimplicialComplex](#), 23
[SolutionsMatDestructive](#), 30
[Source](#), 25, 29
[StandardCocycle](#), 16
[SumOfFpGModules](#), 18
[SumOp](#), 18
[SuspendedChainComplex](#), 7
[SymmetricMatrixToIncidenceMatrix](#), 23
[Syzygy](#), 16

[Target](#), 25, 29
[TensorCentre](#), 12
[TensorWithIntegers](#), 6
[TensorWithIntegersModP](#), 6
[TensorWithRationals](#), 6
[TensorWithTwistedIntegers](#), 6
[TensorWithTwistedIntegersModP](#), 6
[TerminalArrow](#), 25
[TestHap](#), 30
[ThickenedPureCubicalComplex](#), 24
[ThirdHomotopyGroupOfSuspensionB](#), 12
[TietzeReducedResolution](#), 3
[TietzeReduction](#), 17
[TorsionGeneratorsAbelianGroup](#), 14
[TruncatedGComplex](#), 15
[TwistedTensorProduct](#), 3

[UnframeArray](#), 26
[UpperEpicentralSeries](#), 12

[VectorStabilizer](#), 15
[VectorsToFpGModuleWords](#), 18
[VectorsToSymmetricMatrix](#), 23
[ViewPureCubicalComplex](#), 24

[WritePureCubicalComplexAsImage](#), 24

[ZZPersistentHomologyOfPureCubicalComplex](#),
8