

Contents

1	Resolutions of the ground ring	3
2	Resolutions of modules	5
3	Induced equivariant chain maps	6
4	Functors	7
5	Chain complexes	8
6	Homology and cohomology groups	9
7	Poincare series	10
8	Cohomology ring structure	11
9	Cohomology rings of p-groups (mainly $p = 2$)	12
10	Commutator and nonabelian tensor computations	13
11	Lie commutators and nonabelian Lie tensors	14
12	Generators and relators of groups	15
13	Orbit polytopes and fundamental domains	16
14	Cocycles	17
15	Words in free ZG-modules	18
16	FpG-modules	19
17	Meataxe modules	20
18	G-Outer Groups	21
19	Cat-1-groups	22
20	Simplicial groups	23

21	Coxeter diagrams and graphs of groups	26
22	Torsion subcomplexes	27
23	Simplicial Complexes	28
24	Cubical Complexes	29
25	Regular CW-Spaces	31
26	Commutative diagrams and abstract categories	32
27	Arrays and Pseudo lists	33
28	Parallel Computation - Core Functions	34
29	Parallel Computation - Extra Functions	35
30	Some functions for accessing basic data	36
31	Miscellaneous	37

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.
 ResolutionArithmeticGroup("PSL(4,Z)", n) Inputs a positive integer n and one of the following strings:

"SL(2,Z)" , "SL(3,Z)" , "PGL(3,Z[i])" , "PGL(3,Eisenstein_Integers)" , "PSL(4,Z)" , "PSL(4,Z)_b" , "PSL(4,Z)_c" ,

or one of the following strings

"SL(2,Z[sqrt(-2)])" , "SL(2,Z[sqrt(-7)])" , "SL(2,Z[sqrt(-11)])" , "SL(2,Z[sqrt(-19)])" , "SL(2,Z[sqrt(-43)])" , "SL(2,Z[sqrt(-67)])" ,

It returns n terms of a free $\mathbb{Z}G$ -resolution for the group G described by the string. (Subscripts $_b$, $_c$, $_d$ denote alternating subscripts.)

Data for the first list of resolutions was provided by MATHIEU DUTOIR. Data for the second list was provided by MATHIEU DUTOIR.

FreeGResolution(P , n) FreeGResolution(P , n , p) Inputs a non-free $\mathbb{Z}G$ -resolution P with finite stabilizer group G and integer n and p .

ResolutionGTree(P , n) Inputs a non-free $\mathbb{Z}G$ -resolution P of dimension 1 (i.e. a G -tree) with finite stabilizer group G and integer n .

ResolutionSL2Z(p , n) Inputs positive integers m, n and returns n terms of a $\mathbb{Z}G$ -resolution for the group $G = SL(2, \mathbb{Z})$.

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

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

ResolutionAlmostCrystalQuotient(G , n , c) ResolutionAlmostCrystalQuotient(G , n , c , $false$) An almost crystallographic pcg group G and integer n and c .

ResolutionArtinGroup(D , n) Inputs a Coxeter diagram D and an integer $n > 1$. It returns n terms of a free $\mathbb{Z}G$ -resolution for the Artin group $A(D)$.

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 integer n .

ResolutionBieberbachGroup(G) ResolutionBieberbachGroup(G , v) Inputs a torsion free crystallographic group G and integer v .

ResolutionCoxeterGroup(D , n) Inputs a Coxeter diagram D and an integer $n > 1$. It returns k terms of a free $\mathbb{Z}G$ -resolution for the Coxeter group $C(D)$.

ResolutionDirectProduct(R , S) ResolutionDirectProduct(R , S , "internal") Inputs a $\mathbb{Z}G$ -resolution R and a $\mathbb{Z}H$ -resolution S for groups G and H .

ResolutionExtension(g , R , S) ResolutionExtension(g , R , S , "TestFiniteness") ResolutionExtension(g , R , S , n) Inputs a $\mathbb{Z}G$ -resolution R for a group G and a $\mathbb{Z}H$ -resolution S for a group H and integer n .

ResolutionFiniteDirectProduct(R , S) ResolutionFiniteDirectProduct(R , S , "internal") Inputs a $\mathbb{Z}G$ -resolution R and a $\mathbb{Z}H$ -resolution S for groups G and H .

ResolutionFiniteExtension($gensE$, $gensG$, R , n) ResolutionFiniteExtension($gensE$, $gensG$, R , n , $true$) Inputs a $\mathbb{Z}G$ -resolution R for a group G and a list $gensE$ of generators of a group E and integer n .

ResolutionFiniteGroup($gens$, n) ResolutionFiniteGroup($gens$, n , $true$) ResolutionFiniteGroup($gens$, n , $gensG$) Inputs a list $gens$ of generators of a group G and integer n .

ResolutionFiniteSubgroup(R , K) ResolutionFiniteSubgroup(R , $gensG$, $gensK$) Inputs a $\mathbb{Z}G$ -resolution R for a group G and a subgroup K of G .

ResolutionGraphOfGroups(D , n) ResolutionGraphOfGroups(D , n , L) Inputs a graph of groups D and a positive integer n .

ResolutionNilpotentGroup(G , n) ResolutionNilpotentGroup(G , n , "TestFiniteness") Inputs a nilpotent group G and integer n .

ResolutionNormalSeries(L , n) ResolutionNormalSeries(L , n , $true$) ResolutionNormalSeries(L , n , $false$) Inputs a list L of subgroups of a group G and integer n .

ResolutionPrimePowerGroup(P , n) ResolutionPrimePowerGroup(G , n , p) Inputs a p -group P and integer $n > 0$.

ResolutionSmallFpGroup(G , n) ResolutionSmallFpGroup(G , n , p) Inputs a small finitely presented group G and integer n .

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

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 .

TwistedTensorProduct(R , S , $EhomG$, $GmapE$, $NhomE$, $NEhomN$, $EltSE$, $Mult$, $InvE$) Inputs a $\mathbb{Z}G$ -resolution R , a $\mathbb{Z}H$ -resolution S , and a $\mathbb{Z}K$ -resolution E for groups G , H , and K .

ConjugatedResolution(R , x) Inputs a $\mathbb{Z}G$ -resolution R and an element x from some group containing G . It returns a $\mathbb{Z}G$ -resolution for the conjugate group G^x .

RecalculateIncidenceNumbers(R) Inputs a $\mathbb{Z}G$ -resolution R which arises as the cellular chain complex of a regular n -gon.

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

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
`HomToIntegers(X)` Inputs either a ZG -resolution $X = R$, or an equivariant chain map $X = (F : R \longrightarrow S)$. It returns
`HomToIntegersModP(R)` Inputs a ZG -resolution R and returns the cochain complex obtained by applying $HomZG$
`HomToIntegralModule(R,f)` Inputs a ZG -resolution R and a group homomorphism $f : G \longrightarrow GL_n(Z)$ to the group
`TensorWithIntegralModule(R,f)` Inputs a ZG -resolution R and a group homomorphism $f : G \longrightarrow GL_n(Z)$ to the group
`HomToGModule(R,A)` Inputs a ZG -resolution R and an abelian G -outer group A . It returns the G -cocomplex obtained
`InduceScalars(R,hom)` Inputs a ZQ -resolution R and a surjective group homomorphism $hom : G \rightarrow Q$. It returns
`LowerCentralSeriesLieAlgebra(G)` `LowerCentralSeriesLieAlgebra(f)` Inputs a pcg group G . If each qu
`TensorWithIntegers(X)` Inputs either a ZG -resolution $X = R$, or an equivariant chain map $X = (F : R \longrightarrow S)$. It
`FilteredTensorWithIntegers(R)` Inputs a ZG -resolution R for which "filteredDimension" lies in `NamesOfCom`
`TensorWithTwistedIntegers(X,rho)` Inputs either a ZG -resolution $X = R$, or an equivariant chain map $X = (F : R \longrightarrow S)$
`TensorWithIntegersModP(X,p)` Inputs either a ZG -resolution $X = R$, or a characteristics 0 chain complex, or an
`TensorWithTwistedIntegersModP(X,p,rho)` Inputs either a ZG -resolution $X = R$, or an equivariant chain map $X = (F : R \longrightarrow S)$
`TensorWithRationals(R)` Inputs a ZG -resolution R and returns the chain complex obtained by tensoring with the

Chapter 5

Chain complexes

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

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

`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 of X in degree n .

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

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

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

`TensorProductOfChainComplexes(C,D)` Inputs two chain complexes C and D of the same characteristic and returns their tensor product.

`LefschetzNumber(F)` Inputs a chain map $F:C \rightarrow C$ with common source and target. It returns the Lefschetz number of 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 characteristic 0.
`CohomologyModule(C,n)` Inputs a G -cocomplex C together with a non-negative integer n . It returns the cohomology module $H^n(C)$.
`CohomologyPrimePart(C,n,p)` Inputs a cochain complex C in characteristic 0, a positive integer n , and a prime p . It returns the p -part of $H^n(C)$.
`GroupCohomology(X,n)` `GroupCohomology(X,n,p)` Inputs a positive integer n and either a finite group $X = G$ or a nilpotent group $X = G$.
`GroupHomology(X,n)` `GroupHomology(X,n,p)` Inputs a positive integer n and either a finite group $X = G$ or a nilpotent group $X = G$.
`PersistentHomologyOfQuotientGroupSeries(S,n)` `PersistentHomologyOfQuotientGroupSeries(S,n,p)` `PersistentCohomologyOfQuotientGroupSeries(S,n)` `PersistentCohomologyOfQuotientGroupSeries(S,n,p)` Inputs a sequence S of finite groups and a positive integer n .
`UniversalBarCode("UpperCentralSeries",n,d)` `UniversalBarCode("UpperCentralSeries",n,d,k)` Inputs a positive integer n and a prime p .
`PersistentHomologyOfSubGroupSeries(S,n)` `PersistentHomologyOfSubGroupSeries(S,n,p,Resolution)` Inputs a sequence S of finite groups and a positive integer n .
`PersistentHomologyOfFilteredChainComplex(C,n,p)` Inputs a filtered chain complex C (of characteristic 0) and a positive integer n .
`PersistentHomologyOfCommutativeDiagramOfPGroups(D,n)` Inputs a commutative diagram D of finite p -groups and a positive integer n .
`PersistentHomologyOfPureCubicalComplex(L,n,p)` Inputs a positive integer n , a prime p and an increasing chain of subgroups L .
`ZZPersistentHomologyOfPureCubicalComplex(L,n,p)` Inputs a positive integer n , a prime p and any sequence of subgroups L .
`RipsHomology(G,n)` `RipsHomology(G,n,p)` Inputs a graph G , a non-negative integer n (and optionally a prime p).
`BarCode(P)` Inputs an integer persistence matrix P and returns the same information in the form of a binary matrix B .
`BarCodeDisplay(P)` `BarCodeDisplay(P,"mozilla")` Inputs an integer persistence matrix P , and an optional string s .
`Homology(X,n)` Inputs either a chain complex $X = C$ or a chain map $X = (C \longrightarrow D)$. If $X = C$ then the torsion coefficient $H_n(C)$ is returned.
`HomologyPb(C,n)` This is a back-up function which might work in some instances where `Homology(C,n)` fails. It is not recommended.
`HomologyVectorSpace(X,n)` Inputs either a chain complex $X = C$ or a chain map $X = (C \longrightarrow D)$ in prime characteristic p .
`HomologyPrimePart(C,n,p)` Inputs a chain complex C in characteristic 0, a positive integer n , and a prime p . It returns the p -part of $H_n(C)$.
`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 .
`LieAlgebraHomology(A,n)` Inputs a Lie algebra A (over the integers or a field) and a positive integer n . It returns the torsion coefficient $H_n(A)$.
`PrimePartDerivedFunctor(G,R,F,n)` Inputs a finite group G , a positive integer n , at least $n+1$ terms of a ZP -resolution F of R over Z , and a ring R .
`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 $d(n)$ of n .
`RankPrimeHomology(G,n)` Inputs a (smallish) p -group G together with a positive integer n . It returns a function $d(n)$ of n .

Chapter 7

Poincare series

•

`EfficientNormalSubgroups(G)` `EfficientNormalSubgroups(G,k)` Inputs a prime-power group G and, optionally, a prime k . Returns a list of normal subgroups of G .

`ExpansionOfRationalFunction(f,n)` Inputs a positive integer n and a rational function $f(x) = p(x)/q(x)$ where $\deg p < \deg q$. Returns a list of coefficients of the expansion of $f(x)$ as a power series in x .

`PoincareSeries(G,n)` `PoincareSeries(R,n)` `PoincareSeries(L,n)` `PoincareSeries(G)` Inputs a finite group G , a ring R , a lattice L , or a group G . Returns the Poincare series of G .

`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.

`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 compute cup products)
`IntegralRingGenerators(R,n)` Inputs at least $n+1$ terms of a ZG -resolution and integer $n > 0$. It returns a minimal set of generators for the integral cohomology ring.
`ModPCohomologyGenerators(G,n)` `ModPCohomologyGenerators(R)` Inputs either a p -group G and positive integer n , or a finite group G and a prime p dividing $|G|$. It returns a minimal set of generators for the mod p cohomology ring.
`ModPCohomologyRing(G,n)` `ModPCohomologyRing(G,n,level)` `ModPCohomologyRing(R)` `ModPCohomologyRing(R,n)` Inputs either a p -group G and positive integer n , or a finite group G and a prime p dividing $|G|$, or a finite group G and a prime p dividing $|G|$ and a positive integer n , or a finite group G and a prime p dividing $|G|$ and a positive integer n and a positive integer $level$. It returns the mod p cohomology ring.
`ModPRingGenerators(A)` Inputs a mod p cohomology ring A (created using the preceding function). It returns a minimal set of generators for the mod p cohomology ring.
`Mod2CohomologyRingPresentation(G)` `Mod2CohomologyRingPresentation(G,n)` `Mod2CohomologyRingPresentation(G,n,level)` Inputs either a 2-group G and positive integer n , or a finite group G and a prime $p=2$ dividing $|G|$ and a positive integer n , or a finite group G and a prime $p=2$ dividing $|G|$ and a positive integer n and a positive integer $level$. It returns the mod 2 cohomology ring presentation.

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.)

•
| `Mod2CohomologyRingPresentation(G)` `Mod2CohomologyRingPresentation(G,n)` `Mod2CohomologyRingP`
| `PoincareSeriesLHS(G)` 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
`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 $Z^*(G,N)$
`NonabelianExteriorProduct(G,N)` Inputs a finite group G and normal subgroup N . It returns a record E with the f
`NonabelianSymmetricKernel(G)` `NonabelianSymmetricKernel(G,m)` Inputs a finite or nilpotent infinite gro
`NonabelianSymmetricSquare(G)` `NonabelianSymmetricSquare(G,m)` Inputs a finite or nilpotent infinite gro
`NonabelianTensorProduct(G,N)` Inputs a finite group G and normal subgroup N . It returns a record E with the f
`NonabelianTensorSquare(G)` `NonabelianTensorSquare(G,m)` Inputs a finite or nilpotent infinite group G and
`RelativeSchurMultiplier(G,N)` Inputs a finite group G and normal subgroup N . It returns the homology group
`TensorCentre(G)` Inputs a group G and returns the largest central subgroup N such that the induced homomorphis
`ThirdHomotopyGroupOfSuspensionB(G)` `ThirdHomotopyGroupOfSuspensionB(G,m)` Inputs a finite or nilpo
`UpperEpicentralSeries(G,c)` Inputs a nilpotent group G and an integer c . It returns the c -th term of the upper e

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 h

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

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

`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 followi

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

Chapter 12

Generators and relators of groups

•
CayleyGraphOfGroupDisplay(G, X) CayleyGraphOfGroupDisplay($G, X, \text{"mozilla"}$) Inputs a finite group G
IdentityAmongRelatorsDisplay(R, n) IdentityAmongRelatorsDisplay($R, n, \text{"mozilla"}$) Inputs a free Z -
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 with
TorsionGeneratorsAbelianGroup(G) Inputs an abelian group G and returns a generating set $[x_1, \dots, x_n]$ where n

Chapter 13

Orbit polytopes and fundamental domains

`CoxeterComplex(D)` `CoxeterComplex(D,n)` Inputs a Coxeter diagram D of finite type. It returns a non-free ZW-resolution of the Coxeter group $C(D)$.
`ContractibleGcomplex("PSL(4,Z)")` Inputs one of the following strings:

"SL(2,Z)" , "SL(3,Z)" , "PGL(3,Z[i])" , "PGL(3,Eisenstein_Integers)" , "PSL(4,Z)" , "PSL(4,Z)_b" , "PSL(4,Z)_c" ,

or one of the following strings

"SL(2,Z[sqrt(-2)])" , "SL(2,Z[sqrt(-7)])" , "SL(2,Z[sqrt(-11)])" , "SL(2,Z[sqrt(-19)])" , "SL(2,Z[sqrt(-43)])" , "SL(2,Z[sqrt(-47)])" ,

It returns a non-free ZG-resolution for the group G described by the string. The stabilizer groups of cells are finite. (See [13] for details.)

Data for the first list of non-free resolutions was provided provided by MATHIEU DUTOIR. Data for the second list was provided by MATHIEU DUTOIR and JACQUES-LOUIS LANGE.
`QuotientOfContractibleGcomplex(C,D)` Inputs a non-free ZG-resolution C and a finite subgroup D of G which is not contained in any proper parabolic subgroup of G . It returns the non-free ZG-resolution of the quotient group G/D .
`TruncatedGComplex(R,m,n)` Inputs a non-free ZG-resolution R and two positive integers m and n . It returns the non-free ZG-resolution of the truncated resolution R of degree n .
`FundamentalDomainStandardSpaceGroup(v,G)` Inputs a crystallographic group G (represented using AffineCrys) and a rational vector v of length n . It returns the fundamental domain of the standard space group G with respect to the vector v .
`OrbitPolytope(G,v,L)` Inputs a permutation group or matrix group G of degree n and a rational vector v of length n . It returns the orbit polytope of G with respect to the vector v .
`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 . It returns the polytopal complex of G with respect to the vector v .
`PolytopalGenerators(G,v)` Inputs a permutation group or matrix group G of degree n and a rational vector v of length n . It returns the polytopal generators of G with respect to the vector v .
`VectorStabilizer(G,v)` Inputs a permutation group or matrix group G of degree n and a rational vector v of length n . It returns the vector stabilizer of G with respect to the vector v .

Chapter 14

Cocycles

\bullet	
<code>CcGroup(A,f)</code>	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 a G -module A with the standard 2-cocycle f .
<code>CocycleCondition(R,n)</code>	Inputs a resolution R and an integer $n > 0$. It returns an integer matrix M with the following property: $M \cdot \text{cocycles}(R, n) = 0$.
<code>StandardCocycle(R,f,n)</code>	Inputs a resolution R and a standard 2-cocycle $f: G \times G \rightarrow A$. It returns a standard 2-cocycle f on R .
<code>StandardCocycle(R,f,n,q)</code>	Inputs a ZG -resolution R (with contracting homotopy), a positive integer n and an integer q . It returns a standard 2-cocycle f on R .
<code>Syzygy(R,g)</code>	Inputs a ZG -resolution R (with contracting homotopy) and a list $g = [g[1], \dots, g[n]]$ of elements in G . It returns a syzygy s on R .

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 the sum 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 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 relator.

`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 relator 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 Z . It prints the word w in terms of the elements of Z .

`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 at level n .

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 or a list of *FpG*-modules $M[1], M[2], \dots, M[k]$.

`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 P .

`FpGModuleDualBasis(M)` Inputs an *FpG*-module M . It returns a record R with two components: $R.freeModule$ is a free F -module with a basis for M and $R.action$ is a list of matrices representing the action of G on M .

`FpGModuleHomomorphism(M,N,A)` `FpGModuleHomomorphismNC(M,N,A)` Inputs *FpG*-modules M and N over a common group G and a matrix A representing a homomorphism $f: M \rightarrow N$.

`DesuspensionFpGModule(M,n)` `DesuspensionFpGModule(R,n)` Inputs a positive integer n and an *FpG*-module M or a record R as above.

`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 as an *FpG*-module.

`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 as an *FpG*-module.

`IsFpGModuleHomomorphismData(M,N,A)` Inputs *FpG*-modules M and N over a common p -group G . Also inputs a matrix A representing a homomorphism $f: M \rightarrow N$. 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 $G = M$. Returns a list of elements in M .

`ProjectedFpGModule(M,k)` Inputs an *FpG*-module M of ambient dimension $n|G|$, and an integer k between 1 and n . Returns a list of k vectors in M .

`RandomHomomorphismOfFpGModules(M,N)` Inputs two *FpG*-modules M and N over a common group G . It returns a random homomorphism $f: M \rightarrow N$.

`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 as an *FpG*-module.

`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 in G representing the vectors.

Chapter 17

Meataxe modules

•
DesuspensionMtxModule(M) Inputs a meataxe module M over the field of p elements and returns an FpG-module.
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 a

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 $G = E/N$.
`GOuterGroupHomomorphismNC(A,B,phi)` `GOuterGroupHomomorphismNC()` Inputs G -outer groups A and B with common acting group G , and a group homomorphism $\phi: A \rightarrow B$. It returns a G -outer group homomorphism from A to B .
`GOuterHomomorphismTester(A,B,phi)` Inputs G -outer groups A and B with common acting group G , and a group homomorphism $\phi: A \rightarrow B$. It returns a boolean value indicating whether ϕ is a G -outer group homomorphism.
`Centre(A)` Inputs G -outer group A and returns the group theoretic centre of $\text{ActedGroup}(A)$ as a G -outer group.
`DirectProductGog(A,B)` `DirectProductGog(Lst)` Inputs G -outer groups A and B with common acting group G , and a list of G -outer groups. It returns the direct product of the G -outer groups as a G -outer group.

Chapter 19

Cat-1-groups

`AutomorphismGroupAsCatOneGroup(G)` Inputs a group G and returns the Cat-1-group C corresponding to the cross

`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 -

`QuasiIsomorph(C)` Inputs a cat-1-group C and returns a cat-1-group D for which there exists some homomorphism

`ModuleAsCatOneGroup(G,alpha,M)` Inputs a group G , an abelian group M and a homomorphism $\alpha: G \rightarrow \text{Aut}(M)$.

`MooreComplex(C)` Inputs a cat-1-group C and returns its Moore complex as a G -complex (i.e. as a complex of group

`NormalSubgroupAsCatOneGroup(G,N)` Inputs a group G with normal subgroup N . It returns the Cat-1-group C cor

`XmodToHAP(C)` Inputs a cat-1-group C obtained from the Xmod package and returns a cat-1-group D for which `IsHa`

Chapter 20

Simplicial groups

`NerveOfCatOneGroup(G, n)` Inputs a cat-1-group G and a positive integer n . It returns the low-dimensional part of f

This function applies both to cat-1-groups for which `IsHapCatOneGroup(G)` is true, and to cat-1-groups produced us

This function was implemented by VAN LUYEN LE.

`EilenbergMacLaneSimplicialGroup(G, n, dim)` Inputs a group G , a positive integer n , and a positive integer dim

This function was implemented by VAN LUYEN LE.

`EilenbergMacLaneSimplicialGroupMap(f, n, dim)` Inputs a group homomorphism $f : G \rightarrow Q$, a positive integer

This function was implemented by VAN LUYEN LE.

`MooreComplex(G)` Inputs a simplicial group G and returns its Moore complex as a G -complex.

This function was implemented by VAN LUYEN LE.

`ChainComplexOfSimplicialGroup(G)` Inputs a simplicial group G and returns the cellular chain complex C of a C

This function was implemented by VAN LUYEN LE.

`SimplicialGroupMap(f)` Inputs a homomorphism $f : G \rightarrow Q$ of simplicial groups. The function returns an induced

This function was implemented by VAN LUYEN LE.

`HomotopyGroup(G, n)` Inputs a simplicial group G and a positive integer n . The integer n must be less than the length

Representation of elements in the bar resolution For a group G we denote by $B_n(G)$ the free $\mathbb{Z}G$ -modu

We represent a word

$$w = h_1 \cdot [g_{11} | g_{12} | \dots | g_{1n}] - h_2 \cdot [g_{21} | g_{22} | \dots | g_{2n}] + \dots + h_k \cdot [g_{k1} | g_{k2} | \dots | g_{kn}]$$

in $B_n(G)$ as a list of lists:

$$[[+1, h_1, g_{11}, g_{12}, \dots, g_{1n}], [-1, h_2, g_{21}, g_{22}, \dots, g_{2n}] + \dots + [+1, h_k, g_{k1}, g_{k2}, \dots, g_{kn}].$$

`BarResolutionBoundary(w)` This function inputs a word w in the bar resolution module $B_n(G)$ and returns its ima

This function was implemented by VAN LUYEN LE.

`BarResolutionHomotopy(w)` This function inputs a word w in the bar resolution module $B_n(G)$ and returns its ima

This function is currently being implemented by VAN LUYEN LE.

Representation of elements in the bar complex For a group G we denote by $BC_n(G)$ the free abelian group

We represent a word

$$w = [g_{11} | g_{12} | \dots | g_{1n}] - [g_{21} | g_{22} | \dots | g_{2n}] + \dots + [g_{k1} | g_{k2} | \dots | g_{kn}]$$

in $BC_n(G)$ as a list of lists:

$$[[+1, g_{11}, g_{12}, \dots, g_{1n}], [-1, g_{21}, g_{22}, \dots, g_{2n}] + \dots + [+1, g_{k1}, g_{k2}, \dots, g_{kn}].$$

`BarComplexBoundary(w)` This function inputs a word w in the n -th term of the bar complex $BC_n(G)$ and returns its

This function was implemented by VAN LUYEN LE.

`BarResolutionEquivalence(R)` This function inputs a free ZG -resolution R . It returns a component object HE wi

$$equiv(n, -): B_n(G) \rightarrow B_{n+1}(G)$$

satisfying $w - \psi(\phi(w)) = d(n+1, equiv(n, w)) + equiv(n-1, d(n, w))$. where $d(n, -): B_n(G) \rightarrow B_{n-1}(G)$ is the boundar

This function was implemented by VAN LUYEN LE.

`BarComplexEquivalence(R)`

This function inputs a free ZG -resolution R . It first constructs the chain complex $T = \text{TensorWithIntegers}(R)$. The function returns a component object HE with components

- $\text{HE}!.phi(n, w)$ is a function which inputs a non-negative integer n and a word w in $BC_n(G)$. It returns the image of w in T_n under a chain equivalence $\phi: BC_n(G) \rightarrow T_n$.
- $\text{HE}!.psi(n, w)$ is a function which inputs a non-negative integer n and an element w in T_n . It returns the image of w in $BC_n(G)$ under a chain equivalence $\psi: T_n \rightarrow BC_n(G)$.
- $\text{HE}!.equiv(n, w)$ is a function which inputs a non-negative integer n and a word w in $BC_n(G)$. It returns the image of w in $BC_{n+1}(G)$ under a homomorphism $equiv(n, -): BC_n(G) \rightarrow BC_{n+1}(G)$ satisfying

$$w - \psi(\phi(w)) = d(n+1, equiv(n, w)) + equiv(n-1, d(n, w)).$$

where $d(n, -): BC_n(G) \rightarrow BC_{n-1}(G)$ is the boundary homomorphism in the bar complex.

This function was implemented by VAN LUYEN LE.

`Representation of elements in the bar cocomplex`

For a group G we denote by $BC^n(G)$ the free abelian group with basis the lists $[g_1|g_2|\dots|g_n]$ where the g_i range over G .

We represent a word

$$w = [g_{11}|g_{12}|\dots|g_{1n}] - [g_{21}|g_{22}|\dots|g_{2n}] + \dots + [g_{k1}|g_{k2}|\dots|g_{kn}]$$

in $BC^n(G)$ as a list of lists:

$$[[+1, g_{11}, g_{12}, \dots, g_{1n}], [-1, g_{21}, g_{22}, \dots, g_{2n}] + \dots + [+1, g_{k1}, g_{k2}, \dots, g_{kn}].$$

`BarCocomplexCoboundary(w)`

This function inputs a word w in the n -th term of the bar cocomplex $BC^n(G)$ and returns its image under the coboundary homomorphism $d^n: BC^n(G) \rightarrow BC^{n+1}(G)$ in the bar cocomplex.

This function was implemented by VAN LUYEN LE.

Chapter 21

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)` `CoxeterDiagramDisplay(D, "web browser")` Inputs a Coxeter diagram D and displays it in a 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 given by (m_{ij}) where m_{ij} is the order of $s_i s_j$ if $i \neq j$ and $m_{ii} = 2$.

`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 elements of G .

`GraphOfGroupsDisplay(D)` `GraphOfGroupsDisplay(D, "web browser")` Inputs a graph of groups D and displays it in a web browser.

`GraphOfResolutions(D, n)` Inputs a graph of groups D and a positive integer n . It returns a graph of resolutions of D of order n .

`GraphOfGroups(D)` Inputs a graph of resolutions D and returns the corresponding graph of groups.

`GraphOfResolutionsDisplay(D)` Inputs a graph of resolutions D and displays it as a .gif file. It uses the Mozilla browser.

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

`TreeOfGroupsToContractibleGcomplex(D, G)` Inputs a graph of groups D which is a tree, and also inputs the fundamental group G . It returns a contractible G -complex.

`TreeOfResolutionsToContractibleGcomplex(D, G)` Inputs a graph of resolutions D which is a tree, and also inputs the fundamental group G . It returns a contractible G -complex.

#

Chapter 22

Torsion subcomplexes

The torsion subcomplexes subpackage has been conceived and implemented by ALEXANDER D. RAHM.

`IsPnormal(G, p)` Inputs a finite group G and a prime p . Checks if the group G is p -normal for the prime p . Zassenhaus.

`TorsionSubcomplex(groupName, p)` Inputs a cell complex with action of a group. In HAP, presently the following

"SL(2,O[-2])" , "SL(2,O[-7])" , "SL(2,O[-11])" , "SL(2,O[-19])" , "SL(2,O[-43])" , "SL(2,O[-67])" , "SL(2,O[-163])"

where the symbol $O[-m]$ stands for the ring of integers in the imaginary quadratic number field $\mathbb{Q}(\sqrt{-m})$, the latter

The function `TorsionSubcomplex` prints the cells with p -torsion in their stabilizer on the screen and returns the incidence

It is also possible to input the cell complexes

"SL(2,Z)" , "SL(3,Z)" , "PGL(3,Z[i])" , "PGL(3,Eisenstein_Integers)" , "PSL(4,Z)" , "PSL(4,Z)_b" , "PSL(4,Z)_c" ,

provided by MATHIEU DUTOIR, only there will be some warnings printed on the screen regarding the function `reduce`

`DisplayAvailableCellComplexes()`; Displays the cell complexes that are available in HAP.

`VisualizeTorsionSkeleton(groupName, p)` Executes the function `TorsionSubcomplex(groupName, p)` and vi

`ReduceTorsionSubcomplex(groupName, p)` This function may be applied to the cell complexes for which the f

Chapter 23

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 . It 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). It returns the n -th Rips homology group of G .

`Bettinnumbers(T,n)` `Bettinnumbers(T)` Inputs a pure cubical complex, or cubical complex, simplicial complex T and a non-negative integer n . It 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 whose simplices are the non-empty intersections of the d -cubes of T .

`PureComplexToSimplicialComplex(T,k)` Inputs either a d -dimensional pure cubical complex T or a d -dimensional cubical complex T and a non-negative integer k . It returns the simplicial complex of T of dimension at most k .

`RipsChainComplex(G,n)` Inputs a graph G and a non-negative integer n . It returns $n+1$ terms of a chain complex whose homology is the n -th Rips homology group of G .

`VectorsToSymmetricMatrix(M)` `VectorsToSymmetricMatrix(M,distance)` Inputs a matrix M of rational numbers and a non-negative integer $distance$. It returns a symmetric matrix whose entries are the minimum of the entries of M over all paths of length at most $distance$ between the vertices of the matrix.

`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. It 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 . It 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 $n=0$ it returns the number of path components of S . If $n>0$ it 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)` `SymmetricMatrixToIncidenceMatrix(S,t,d)` Inputs a symmetric 0/1 matrix M and a non-negative integer t (and optionally a non-negative integer d). It returns a symmetric 0/1 matrix whose entries are the number of t -simplices of the simplicial complex of M that contain the d -simplex.

`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 $M_{ij}=1$.

`CayleyGraphOfGroup(G,A)` Inputs a group G and a set A of generators. It returns the Cayley graph.

`PathComponentsOfGraph(G,n)` Inputs a graph G and a nonnegative integer n . If $n=0$ it returns the number of path components of G . If $n>0$ it 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 and G' have the same homology.

`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 $f:K \rightarrow L$. It 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 24

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.
`ReadLinkImageAsPureCubicalComplex("file.png")` Reads a link image file ("file.png") 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 cubical complex, or pure cubical complex T .
`Dimension(T)` This returns the dimension of the cubical complex, or pure cubical complex T .
`WritePureCubicalComplexAsImage(T,"filename","ext")` Inputs a 2-dimensional pure cubical complex T , and a filename and extension. It writes the image to the file.
`ViewPureCubicalComplex(T)` `ViewPureCubicalComplex(T,"mozilla")` Inputs a 2-dimensional pure cubical complex T , and an optional browser name. It opens a window showing the image.
`Homology(T,n)` `Homology(T)` Inputs a pure cubical complex, or cubical complex, or simplicial complex T and a non-negative integer n . It returns the n -th homology group.
`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 . It returns the n -th Betti number.
`DirectProductOfPureCubicalComplexes(M,N)` Inputs two pure cubical complexes M, N and returns their direct product.
`SuspensionOfPureCubicalComplex(M)` Inputs a pure cubical complex M and returns a pure cubical complex with one more dimension.
`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 $1, \dots, \dim T$. It returns the n -th 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 `ContractPureCubicalComplex(T)`.
`ZigZagContractedPureCubicalComplex(T)` Inputs a pure cubical complex T and returns a homotopy equivalent complex.
`ContractCubicalComplex(T)` Inputs a cubical complex T and removes cells without changing the homotopy type.
`DVFRducedCubicalComplex(T)` Inputs a cubical complex T and returns a non-regular cubical complex R by construction.
`SkeletonOfCubicalComplex(T,n)` Inputs a cubical complex, or pure cubical complex T and positive integer n . It returns the n -skeleton.
`ContractibleSubomplexOfPureCubicalComplex(T)` Inputs a pure cubical complex T and returns a maximal contractible subcomplex.
`AcyclicSubomplexOfPureCubicalComplex(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 pure cubical complex S . It returns a maximal subcomplex of T homotopy equivalent to S .
`HomotopyEquivalentMinimalPureCubicalSubcomplex(T,S)` Inputs a pure cubical complex T together with a pure cubical complex S . It returns a minimal subcomplex of T 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 euclidean neighborhood retract.
`CropPureCubicalComplex(T)` Inputs a pure cubical complex T and returns a pure cubical complex S obtained from T by cropping.
`BoundingPureCubicalComplex(T)` Inputs a pure cubical complex T and returns a contractible pure cubical complex bounding T .
`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 an optional boolean. It returns the Morse filtration.
`ComplementOfPureCubicalComplex(T)` Inputs a pure cubical complex T and returns a pure cubical complex S . A euclidean neighborhood retract.
`PureCubicalComplexToTextFile(file,M)` Inputs a pure cubical complex M and a string containing the address of a text file. It writes the complex to the file.

Chapter 25

Regular CW-Spaces

`SimplicialComplexToRegularCWSpace(K)` Inputs a simplicial complex K and returns the corresponding regular CW-space W whose cells are the simplices of K .
`CubicalComplexToRegularCWSpace(K)` Inputs a pure cubical complex (or cubical complex) K and returns the corresponding regular CW-space W whose cells are the cubes of K .
`CriticalCellsOfRegularCWSpace(Y)` Inputs a regular CW-space Y and returns the critical cells of Y with respect to the standard CW-structure.
`ChainComplex(Y)` Inputs a regular CW-space Y and returns the cellular chain complex of a CW-space W whose cells are the cells of Y .
`ChainComplexOfRegularCWSpace(Y)` Inputs a regular CW-space Y and returns the cellular chain complex of Y .
`FundamentalGroup(Y)` `FundamentalGroup(Y,n)` Inputs a regular CW-space Y and, optionally, the number of generators n and returns the fundamental group of Y .

Chapter 26

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 the mappings is the identity mapping.
`NormalSeriesToQuotientDiagram(L)` `NormalSeriesToQuotientDiagram(L,M)` Inputs an increasing (or decreasing) normal series L of a group G and returns the quotient diagram ND corresponding to L .
`NerveOfCommutativeDiagram(D)` Inputs a commutative diagram D and returns the commutative diagram ND corresponding to D .
`GroupHomologyOfCommutativeDiagram(D,n)` `GroupHomologyOfCommutativeDiagram(D,n,prime)` Group homology of a commutative diagram D of finite p -groups.
`PersistentHomologyOfCommutativeDiagramOfPGroups(D,n)` Inputs a commutative diagram D of finite p -groups and returns the persistent homology of D .

ABSTRACT CATEGORIES

`CategoricalEnrichment(X,Name)` Inputs a structure X such as a group or group homomorphism, together with the name of a 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 27

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 (and f to each entry of A).

`PermuteArray(A,f)` Inputs an array A of dimension d and a permutation f of degree at most d . It returns the array obtained by permuting the entries of A according to f .

`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 arrays of 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 arrays of dimension d .

`ArrayIterate(d)` Inputs a positive integer d and returns a function `ArrayIt(Dimensions,f)`. This function inputs a list of dimension d and returns f applied to each entry of the list.

`BinaryArrayToTextFile(file,A)` Inputs a string containing the address of a file, and an array A of 0s and 1s. The file is created if it does not exist. The contents of the file are the binary representation of the entries of A .

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

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

`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 entries of K to the end of L .

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

Parallel Computation - Core Functions

- 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

`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 c

`ChildGet("A", s)` This functions 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 i

Chapter 29

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 30

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 $C!.dimension$.
`EvaluateProperty(X,"name")` Inputs a component object X (such as a ZG -resolution or chain map) and a string `name`.
`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 map itself).
`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 31

Miscellaneous

`SL2Z(p)` `SL2Z(1/m)` Inputs a prime p or the reciprocal $1/m$ of a square free integer m . In the first case the function returns the modular group $SL(2, \mathbb{Z}/p\mathbb{Z})$. In the second case it returns the congruence subgroup $\Gamma_0(m)$.

`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 such that $|G/L_i| \leq n$.

`Classify(L,Inv)` Inputs a list of objects L and a function Inv which computes an invariant of each object. It returns a list of objects C such that $Inv(C) = Inv(L)$ and C is a refinement of L .

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

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

`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 C such that $CM = B$.

`LinearHomomorphismsPersistenceMat(L)` Inputs a composable sequence L of vector space homomorphisms. It returns a matrix M such that $M_{ij} = \dim \ker L_i \cap \text{Im } L_j$.

`NormalSeriesToQuotientHomomorphisms(L)` Inputs an (increasing or decreasing) chain L of normal subgroups of a group G . It returns a sequence of homomorphisms $f_i : G/L_i \longrightarrow G/L_{i+1}$.

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

Index

AcyclicSubomplexOfPureCubicalComplex, 30
Add, 33
AddFreeWords, 18
AddFreeWordsModP, 18
AlgebraicReduction, 18
Append, 33
Array, 33
ArrayAssign, 33
ArrayAssignFunctions, 33
ArrayDimension, 33
ArrayDimensions, 33
ArrayIterate, 33
ArraySum, 33
ArrayToPureCubicalComplex, 30
ArrayValue, 33
ArrayValueFunctions, 33
AutomorphismGroupAsCatOneGroup, 22

BaerInvariant, 13
Bar Cocomplex, 25
Bar Complex, 24
Bar Resolution, 24
BarCocomplexCoboundary, 25
BarCode, 9
BarCodeDisplay, 9
BarComplexBoundary, 24
BarComplexEquivalence, 25
BarResolutionBoundary, 24
BarResolutionEquivalence, 24
BarResolutionHomotopy, 24
Bettinnumbers, 28, 30
BigStepLCS, 37
BinaryArrayToTextFile, 33
BoundaryMap, 36
BoundaryMatrix, 36
BoundaryOfPureCubicalComplex, 30
BoundingPureCubicalComplex, 30

CategoricalEnrichment, 32
CategoryName, 32
CayleyGraphOfGroup, 28
CayleyGraphOfGroupDisplay, 15
CcGroup (HAPcocyclic), 17
CechComplexOfPureCubicalComplex, 28
Centre, 21
ChainComplex, 8, 31
ChainComplexOfPair, 8
ChainComplexOfRegularCWSpace, 31
ChainComplexOfSimplicialGroup, 24
ChainInclusionOfPureCubicalPair, 30
ChainMapOfPureCubicalPairs, 30
ChainMapOfSimplicialMap, 28
ChevalleyEilenbergComplex, 8
ChildClose, 34
ChildCommand, 34
ChildFunction, 35
ChildGet, 34
ChildProcess, 34
ChildPut, 34
ChildRead, 35
ChildReadEval, 35
Classify, 37
Coclass, 13
CocycleCondition, 17
Cohomology, 9
CohomologyModule, 9
CohomologyPrimePart, 9
ComplementOfPureCubicalComplex, 30
Compose(f,g), 37
CompositionSeriesOfFpGModules, 19
ConjugatedResolution, 4
ContractCubicalComplex, 30
ContractedComplex, 30
ContractGraph, 28
ContractibleGcomplex, 16
ContractibleSubcomplexOfSimplicialComplex, 28

- ContractibleSubomplexOfPureCubicalComplex, 30
- ContractPureCubicalComplex, 30
- CoreducedChainComplex, 8
- CoxeterComplex, 16
- CoxeterDiagramComponents, 26
- CoxeterDiagramDegree, 26
- CoxeterDiagramDisplay, 26
- CoxeterDiagramFpArtinGroup, 26
- CoxeterDiagramFpCoxeterGroup, 26
- CoxeterDiagramIsSpherical, 26
- CoxeterDiagramMatrix, 26
- CoxeterDiagramVertices, 26
- CoxeterSubDiagram, 26
- CriticalCellsOfRegularCWSpace, 31
- CropPureCubicalComplex, 30
- CubicalComplexToRegularCWSpace, 31
- DesuspensionFpGModule, 19
- DesuspensionMtxModule, 20
- Dimension, 36
- DirectProductGog, 21
- DirectProductOfPureCubicalComplexes, 30
- DirectSumOfFpGModules, 19
- DisplayAvailableCellComplexes, 27
- DVFRducedCubicalComplex, 30
- EilenbergMacLaneSimplicialGroup, 24
- EilenbergMacLaneSimplicialGroupMap, 24
- EpiCentre, 13
- EquivariantChainMap, 6
- EulerCharacteristic, 28
- EvaluateProperty, 36
- EvenSubgroup, 26
- ExpansionOfRationalFunction, 10
- ExtendScalars, 7
- FilteredTensorWithIntegers, 7
- FpGModule, 19
- FpGModuleDualBasis, 19
- FpGModuleHomomorphism, 19
- FpG_to_MtxModule, 20
- FrameArray, 33
- FreeGResolution, 4
- FundamentalDomainStandardSpaceGroup (HAPcryst), 16
- FundamentalGroup, 31
- FundamentalGroupOfRegularCWSpace, 31
- GeneratorsOfFpGModule, 19
- GeneratorsOfMtxModule, 20
- GOuterGroup, 21
- GOuterGroupHomomorphismNC, 21
- GOuterHomomorphismTester, 21
- GraphDisplay, 28
- GraphOfGroups, 26
- GraphOfGroupsDisplay, 26
- GraphOfGroupsTest, 26
- GraphOfResolutions, 26
- GraphOfResolutionsDisplay, 26
- GraphOfSimplicialComplex, 28
- GroupAlgebraAsFpGModule, 19
- GroupCohomology, 9
- GroupHomology, 9
- GroupHomologyOfCommutativeDiagram, 32
- GroupOfResolution, 36
- HAPcopyright, 37
- HAPPrintTo, 34
- HAPRead, 34
- HasInitialObject, 32
- HasTerminalObject, 32
- Homology, 9, 30
- HomologyPb, 9
- HomologyPrimePart, 9
- HomologyVectorSpace, 9
- HomomorphismChainToCommutativeDiagram, 32
- HomotopyEquivalentMaximalPureCubicalSubcomplex, 30
- HomotopyEquivalentMinimalPureCubicalSubcomplex, 30
- HomotopyGroup, 22, 24
- HomotopyModule, 22
- HomToGModule, 7
- HomToIntegers, 7
- HomToIntegersModP, 7
- HomToIntegralModule, 7
- IdentityAmongRelatorsDisplay, 15
- IdentityArrow, 32
- ImageOfFpGModuleHomomorphism, 19
- IncidenceMatrixToGraph, 28
- InduceScalars, 7
- InitialArrow, 32
- IntegralCupProduct, 11

- IntegralRingGenerators, 11
- IntersectionOfFpGModules, 19
- IsAspherical, 15
- IsAvailableChild, 34
- IsCategoryArrow, 32
- IsCategoryObject, 32
- IsFpGModuleHomomorphismData, 19
- IsLieAlgebraHomomorphism, 37
- IsPnormal, 27
- IsSuperperfect, 37
- LefschetzNumber, 8
- LeibnizAlgebraHomology, 9
- LeibnizComplex, 8
- LeibnizQuasiCoveringHomomorphism, 14
- Length, 36
- LieAlgebraHomology, 9
- LieCoveringHomomorphism, 14
- LieEpiCentre, 14
- LieExteriorSquare, 14
- LieTensorCentre, 14
- LieTensorSquare, 14
- LinearHomomorphismsPersistenceMat, 37
- ListToPseudoList, 33
- LowerCentralSeriesLieAlgebra, 7
- MakeHAPManual, 37
- Map, 36
- Mapping, 32
- MaximalSimplicesToSimplicialComplex, 28
- MaximalSubmoduleOfFpGModule, 19
- MaximalSubmodulesOfFpGModule, 19
- Mod2CohomologyRingPresentation (HAP-prime), 12
- ModPCohomologyGenerators, 11
- ModPCohomologyRing, 11
- ModP RingGenerators, 11
- ModuleAsCatOneGroup, 22
- MooreComplex, 22, 24
- MorseFiltration, 30
- MultipleOfFpGModule, 19
- MultiplyWord, 18
- Negate, 18
- NegateWord, 18
- NerveOfCatOneGroup, 24
- NerveOfCommutativeDiagram, 32
- NextAvailableChild, 34
- NonabelianExteriorProduct, 13
- NonabelianSymmetricKernel, 13
- NonabelianSymmetricSquare, 13
- NonabelianTensorProduct, 13
- NonabelianTensorSquare, 13
- NormalSeriesToQuotientDiagram, 32
- NormalSeriesToQuotientHomomorphisms, 37
- NormalSubgroupAsCatOneGroup, 22
- Object, 32
- OrbitPolytope, 16
- ParallelList, 35
- PathComponentOfPureCubicalComplex, 30
- PathComponentsOfGraph, 28
- PathComponentsOfSimplicialComplex, 28
- PermToMatrixGroup, 37
- PermuteArray, 33
- PersistentCohomologyOfQuotientGroupSeries, 9
- PersistentHomologyOfCommutativeDiagramOfPGroups, 32
- PersistentHomologyOfFilteredChainComplex, 9
- PersistentHomologyOfPureCubicalComplex, 9
- PersistentHomologyOfQuotientGroupSeries, 9
- PersistentHomologyOfSubGroupSeries, 9
- PoincareSeries, 10
- PoincareSeriesLHS (HAPprime), 12
- PoincareSeriesPrimePart, 10
- PolytopalComplex, 16
- PolytopalGenerators, 16
- Prank, 10
- PresentationOfResolution, 15
- PrimePartDerivedFunctor, 9
- PrintZGword, 18
- ProjectedFpGModule, 19
- PureComplexToSimplicialComplex, 28
- PureCubicalComplex, 30
- PureCubicalComplexDifference, 30
- PureCubicalComplexIntersection, 30
- PureCubicalComplexToTextFile, 30
- PureCubicalComplexUnion, 30
- QuasiIsomorph, 22
- QuillenComplex, 28
- QuotientOfContractibleGcomplex, 16
- RadicalOfFpGModule, 19
- RadicalSeriesOfFpGModule, 19

- RandomHomomorphismOfFpGModules, 19
- Rank, 19
- RankHomologyPGroup, 9
- RankPrimeHomology, 9
- ReadImageAsPureCubicalComplex, 30
- ReadImageSequenceAsPureCubicalComplex, 30
- ReadLinkImageAsPureCubicalComplex, 30
- RecalculateIncidenceNumbers, 4
- ReducedSuspendedChainComplex, 8
- ReduceTorsionSubcomplex, 27
- RefineClassification, 37
- RelativeSchurMultiplier, 13
- ResolutionAbelianGroup, 4
- ResolutionAlmostCrystalGroup, 4
- ResolutionAlmostCrystalQuotient, 4
- ResolutionArithmeticGroup, 4
- ResolutionArtinGroup, 4
- ResolutionAsphericalPresentation, 4
- ResolutionBieberbachGroup (HAPcryst), 4
- ResolutionBoundaryOfWord, 18
- ResolutionCoxeterGroup, 4
- ResolutionDirectProduct, 4
- ResolutionExtension, 4
- ResolutionFiniteDirectProduct, 4
- ResolutionFiniteExtension, 4
- ResolutionFiniteGroup, 4
- ResolutionFiniteSubgroup, 4
- ResolutionFpGModule, 5
- ResolutionGraphOfGroups, 4
- ResolutionGTree, 4
- ResolutionNilpotentGroup, 4
- ResolutionNormalSeries, 4
- ResolutionPrimePowerGroup, 4
- ResolutionSL2Z, 4
- ResolutionSmallFpGroup, 4
- ResolutionSubgroup, 4
- ResolutionSubnormalSeries, 4
- RipsChainComplex, 28
- RipsHomology, 9
- SimplicialComplexToRegularCWSpace, 31
- SimplicialGroupMap, 24
- SimplicialMap, 28
- SimplicialMapNC, 28
- SimplicialNerveOfGraph, 28
- SingularitiesOfPureCubicalComplex, 30
- SkeletonOfCubicalComplex, 30
- SkeletonOfSimplicialComplex, 28
- SL2Z, 37
- SolutionsMatDestructive, 37
- Source, 32, 36
- StandardCocycle, 17
- SumOfFpGModules, 19
- SumOp, 19
- SuspendedChainComplex, 8
- SuspensionOfPureCubicalComplex, 30
- SymmetricMatrixToIncidenceMatrix, 28
- Syzygy, 17
- Target, 32, 36
- TensorCentre, 13
- TensorProductOfChainComplexes, 8
- TensorWithIntegers, 7
- TensorWithIntegersModP, 7
- TensorWithIntegralModule, 7
- TensorWithRationals, 7
- TensorWithTwistedIntegers, 7
- TensorWithTwistedIntegersModP, 7
- TerminalArrow, 32
- TestHap, 37
- ThickenedPureCubicalComplex, 30
- ThirdHomotopyGroupOfSuspensionB, 13
- TietzeReducedResolution, 4
- TietzeReduction, 18
- TorsionGeneratorsAbelianGroup, 15
- TorsionSubcomplex, 27
- TreeOfGroupsToContractibleGcomplex, 26
- TreeOfResolutionsToContractibleGcomplex, 26
- TruncatedGComplex, 16
- TwistedTensorProduct, 4
- UnframeArray, 33
- UniversalBarCode, 9
- UpperEpicentralSeries, 13
- VectorStabilizer, 16
- VectorsToFpGModuleWords, 19
- VectorsToSymmetricMatrix, 28
- ViewPureCubicalComplex, 30
- VisualizeTorsionSkeleton, 27
- WritePureCubicalComplexAsImage, 30
- XmodToHAP, 22
- ZigZagContractedPureCubicalComplex, 30

ZZPersistentHomologyOfPureCubicalComplex,
9