Contents

Chapter 1

Resolutions of the ground ring

`ResolutionAbelianGroup(L,n)` `ResolutionAbelianGroup(G,n)` Inputs a list $L := [m_1, m_2, ..., m_d]$ of nonneg
`ResolutionAlmostCrystalGroup(G,n)` Inputs a positive integer n and an almost crystallographic pcp group G. I
`ResolutionAlmostCrystalQuotient(G,n,c)` `ResolutionAlmostCrystalQuotient(G,n,c,false)` An alm
`ResolutionArtinGroup(D,n)` Inputs a Coxeter diagram D and an integer $n > 1$. It returns n terms of a free ZG-re
`ResolutionAsphericalPresentation(F,R,n)` Inputs a free group F, a set R of words in F which constitute an a
`ResolutionBieberbachGroup( G )`
`ResolutionBieberbachGroup( G, v )` Inputs a Bieberbach group G (represented using AffineCrystGroupOnR
`ResolutionDirectProduct(R,S)` `ResolutionDirectProduct(R,S,"internal")` Inputs a ZG-resolution R
`ResolutionExtension(g,R,S)` `ResolutionExtension(g,R, S,"TestFiniteness")` `ResolutionExte`
`ResolutionFiniteDirectProduct(R,S)` `ResolutionFiniteDirectProduct(R,S, "internal")` Inputs a
`ResolutionFiniteExtension(gensE,gensG,R,n)` `ResolutionFiniteExtension(gensE,gensG,R,n,true`
`ResolutionFiniteGroup(gens,n)` `ResolutionFiniteGroup(gens,n,true)` `ResolutionFiniteGroup(g`
`ResolutionFiniteSubgroup(R,K)` `ResolutionFiniteSubgroup(R,gensG,gensK)` Inputs a ZG-resolution fo
`ResolutionGraphOfGroups(D,n)` `ResolutionGraphOfGroups(D,n,L)` Inputs a graph of groups D and a p
`ResolutionNilpotentGroup(G,n)` `ResolutionNilpotentGroup(G,n,"TestFiniteness")` Inputs a nilpot
`ResolutionNormalSeries(L,n)` `ResolutionNormalSeries(L,n,true)` `ResolutionNormalSeries(L,n`
`ResolutionPrimePowerGroup(P,n)` `ResolutionPrimePowerGroup(G,n,p)` Inputs a p-group P and integer
`ResolutionSmallFpGroup(G,n)` `ResolutionSmallFpGroup(G,n,p)` Inputs a small finitely presented grou
`ResolutionSubgroup(R,K)` Inputs a ZG-resolution for an (infinite) group G and a subgroup K of finite index $|G:$
`ResolutionSubnormalSeries(L,n)` Inputs a positive integer n and a list $L = [L_1, \ldots, L_k]$ of subgroups L_i of a fir
`TwistedTensorProduct(R,S,EhomG,GmapE,NhomE,NEhomN,EltsE,Mult,InvE)` Inputs a ZG-resolution R, a ZN

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 $f : G -$

Chapter 4

Functors

•

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

`HomToIntegersModP(R)` Inputs a ZG-resolution R and returns the cochain complex obtained by applying $HomZC$

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

`LowerCentralSeriesLieAlgebra(G)` `LowerCentralSeriesLieAlgebra(f)` Inputs a pcp group G. If each

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

`TensorWithIntegersModP(X,p)` Inputs either a ZG-resolution $X = R$, or an equivariant chain map $X = (F : R -$

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

Chapter 5

Chain complexes

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

Chapter 6

Homology and cohomology groups

`Cohomology(X,n)` Inputs either a cochain complex $X = C$ or a cochain map $X = (C \longrightarrow D)$ over the integers Z tog
`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 $X = G$ o
`GroupHomology(X,n)`
`GroupHomology(X,n,p)` Inputs a positive integer n and either a finite group $X = G$ or a Coxeter diagram $X = D$ rep
`Homology(X,n)` Inputs either a chain complex $X = C$ or a chain map $X = (C \longrightarrow D)$.If $X = C$ then the torsion coeffi
`HomologyPb(C,n)` This is a back-up function which might work in some instances where $Homology(C,n)$ fails. It is
`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 Z or over a field K
`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-reso
`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
`ExpansionOfRationalFunction(f,n)` Inputs a positive integer n and a rational function $f(x) = p(x)/q(x)$ where t
`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 $f(x)$ =
`PoincareSeriesPrimePart(G,p,n)` Inputs a finite group G, a prime p, and a positive integer n. It returns a quotie
`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 availabl
`IntegralRingGenerators(R,n)` Inputs at least $n+1$ terms of a ZG-resolution and integer $n>0$. It returns a min
`ModPCohomologyGenerators(G,n)`
`ModPCohomologyGenerators(R)` Inputs either a p-group G and positive integer n, or else n terms of a minimal Z
`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 Z
`ModPRingGenerators(A)` Inputs a mod p cohomology ring A (created using the preceeding function). It returns a

Chapter 9

Commutator and nonabelian tensor computations

•

`BaerInvariant(G,c)` Inputs a nilpotent group G and integer c>0. It returns the Baer invariant $M^(c)(G)$ defined a
`Coclass(G)` Inputs a group G of prime-power order p^n and nilpotency class c say. It returns the integer $r = n - c$
`EpiCentre(G,N)`
`EpiCentre(G)` Inputs a finite group G and normal subgroup N and returns a subgroup $Z^*(G,N)$ of the centre of N.
`NonabelianExteriorProduct(G,N)` Inputs a finite group G and normal subgroup N. It returns a record E with t
`NonabelianSymmetricKernel(G)`
`NonabelianSymmetricKernel(G,m)` Inputs a finite or nilpotent infinite group G and returns the abelian invariant
`NonabelianSymmetricSquare(G)`
`NonabelianSymmetricSquare(G,m)` Inputs a finite or nilpotent infinite group G and returns a record T with the
`NonabelianTensorProduct(G,N)` Inputs a finite group G and normal subgroup N. It returns a record E with the
`NonabelianTensorSquare(G)`
`NonabelianTensorSquare(G,m)` Inputs a finite or nilpotent infinite group G and returns a record T with the follo
`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 homomorph
`ThirdHomotopyGroupOfSuspensionB(G)`
`ThirdHomotopyGroupOfSuspensionB(G,m)` Inputs a finite or nilpotent infinite group G and returns the abelian i
`UpperEpicentralSeries(G,c)` Inputs a nilpotent group G and an integer c. It returns the c-th term of the upper

Chapter 10

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 follo
  `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 11

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 $[x_1, \ldots, x_n]$ where

Chapter 12

Orbit polytopes and fundamental domains

•

`FundamentalDomainAffineCrystGroupOnRight(v,G)` Inputs a crystallographic group G (represented using Affi
`OrbitPolytope(G,v,L)` Inputs a permutation group or matrix group G of degree n and a rational vector v of leng
`PolytopalComplex(G,v)`
`PolytopalComplex(G,v,n)` Inputs a permutation group or matrix group G of degree n and a rational vector v of
`PolytopalGenerators(G,v)` Inputs a permutation group or matrix group G of degree n and a rational vector v o
`VectorStabilizer(G,v)` Inputs a permutation group or matrix group G of degree n and a rational vector of degr

Chapter 13

Cocycles

•

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

Chapter 14

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
`AddFreeWordsModP(v,w,p)` Inputs two words v, w in a free ZG-module and the characteristic p of Z. It returns th
`AlgebraicReduction(w)`
`AlgebraicReduction(w,p)` Inputs a word w in a free ZG-module and returns a reduced version of the word in w
`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 t
`TietzeReduction(S,w)` Inputs a set S of words in a free ZG-module, and a word w in the module. The function r

Chapter 15

FpG-modules

`DirectSumOfFpGModules(M,N)`
`DirectSumOfFpGModules([ M[1], M[2], ..., M[k] ]))` Inputs two FpG-modules M and N with common g
`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 ret
`FpGModuleDualBasis(M)` Inputs an FpG-module M. It returns a record R with two components:$R.freeModule$ i
`FpGModuleHomomorphism(M,N,A)`
`FpGModuleHomomorphismNC(M,N,A)` Inputs FpG-modules M and N over a common p-group G. Also inputs a lis
`DesuspensionFpGModule(M,n)`
`DesuspensionFpGModule(R,n)` Inputs a positive integer n and and FpG-module M. It returns an FpG-module D'
`RadicalOfFpGModule(M)` Inputs an FpG-module M with G a p-group, and returns the Radical of M as an FpG-
`GeneratorsOfFpGModule(M)` Inputs an FpG-module M and returns a matrix whose rows correspond to a minima
`ImageOfFpGModuleHomomorphism(f)` Inputs an FpG-module homomorphism $f: M \longrightarrow N$ and returns its image
`IntersectionOfFpGModules(M,N)` Inputs two FpG-modules M,N arising as submodules in a common free mod
`IsFpGModuleHomomorphismData(M,N,A)` Inputs FpG-modules M and N over a common p-group G. Also input
`MultipleOfFpGModule(w,M)` Inputs an FpG-module M and a list $w := [g_1, ..., g_t]$ of elements in the group $G = M$
`ProjectedFpGModule(M,k)` Inputs an FpG-module M of ambient dimension $n|G|$, and an integer k between 1 an
`RandomHomomorphismOfFpGModules(M,N)` Inputs two FpG-modules M and N over a common group G. It retur
`Rank(f)` Inputs an FpG-module homomorphism $f: M \longrightarrow N$ and returns the dimension of the image of f as a ve
`SumOfFpGModules(M,N)` Inputs two FpG-modules M,N arising as submodules in a common free module $(FG)^n$
`SumOp(f,g)` Inputs two FpG-module homomorphisms $f,g: M \longrightarrow N$ with common sorce and common target. It
`VectorsToFpGModuleWords(M,L)` Inputs an FpG-module M and a list $L = [v_1, \ldots, v_k]$ of vectors in M. It returns

Chapter 16

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 17

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.
`DirectProduct(A,B)`
`DirectProduct(Lst)` Inputs G-outer groups A and B with common acting group, and returns their group-theoretic d

Chapter 18

Cat-1-groups

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

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

`ModuleAsCatOneGroup(G,alpha,M)` Inputs a group G, an abelian group M and a homomorphism $\alpha : G \to Aut(M)$

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

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

Chapter 19

Coxeter diagrams and graphs of groups

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

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

`CoxeterDiagramDisplay(D)`

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

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

`CoxeterDiagramFpCoxeterGroup(D)` Inputs a Coxeter diagram D and returns the corresponding finitely present

`CoxeterDiagramIsSpherical(D)` Inputs a Coxeter diagram D and returns "true" if the associated Coxeter group

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

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

`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 x

`GraphOfGroupsDisplay(D)`

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

`GraphOfGroupsTest(D)` Inputs an object D and itries to test whether it is a Graph of Groups. However, it DOES

Chapter 20

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 o
`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
`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 oppo
`Source(f)` Inputs a chain map, or cochain map, or equivariant chain map, or FpG-module homomorphism f and
`Target(f)` Inputs a chain map, or cochain map, or equivariant chain map, or FpG-module homomorphism f and

Chapter 21

Parallel Computation - Core Functions

`ChildProcess()`
`ChildProcess("computer.ac.wales")` This starts a GAP session as a child process and returns a stream to the ch

- 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 fal
`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 functions copies a GAP object A on the child process s and returns it on the parent process

Chapter 22

Parallel Computation - Extra Functions

`ChildFunction("function(arg);",s)` This runs the GAP function "function(arg);" on a child process accessed b
`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("functi
`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 23

Topological Data Analysis

`MatrixToTopologicalSpace(A,n)` Inputs an integer matrix A and an integer n. It returns a 2-dimensional topolog
`ReadImageAsTopologicalSpace("file.png",n)` `ReadImageAsTopologicalSpace("file.png",[m,n])` Rea
`ReadImageAsMatrix("file.png")` Reads an image file (”file.png”, ”file.eps”, ”file.bmp” etc) and returns an integ
`WriteTopologicalSpaceAsImage(T,"filename","ext")` Inputs a 2-dimensional topological space T, and a file
`ViewTopologicalSpace(T)` `ViewTopologicalSpace(T,"mozilla")` Inputs a topological space T, and optional
`Bettinumbers(T,n)` `Bettinumbers(T)` Inputs a topological space T and a non-negative integer n. It returns the
`PathComponent(T,n)` Inputs a topological space T and an integer n in the rane 0, ..., Bettinumbers(T,0) . It returns
`SingularChainComplex(A)` Inputs a topological space T and returns a (usually very large) integral chain complex
`ContractTopologicalSpace(T)` Inputs a topological space T of dimension d and removes d-dimensional cells fr
`BoundaryTopologicalSpace(T)` Inputs a topological space T and returns its boundars as a topological space.
`BoundarySingularities(T)` Inputs a topological space T and returns the subspace of points in the boundary whe
`ThickenedTopologicalSpace(T)` `ThickenedTopologicalSpace(T,n)` Inputs a topological space T and return
`ComplementTopologicalSpace(T)` Inputs a topological space T and returns a topological space S. A euclidean po
`ConcatenatedTopologicalSpace(L)` Inputs a list L of topological spaces whose underlying arrays of numbers al

Chapter 24

Pseudo lists

`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 th
`Append(L,K)` Let L be a pseudo list and K a list whose objects are compatible with those in L. This operation appli
`ListToPseudoList(L)` Inputs a list L and returns the pseudo list representation of L.

Chapter 25

Miscellaneous

`BigStepLCS(G,n)` Inputs a group G and a positive integer n. It returns a subseries $G = L_1 > L_2 > \ldots L_k = 1$ of the l
`Compose(f,g)` Inputs two FpG-module homomorphisms $f : M \longrightarrow N$ and $g : L \longrightarrow M$ with $Source(f) = Target($
`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 a
`IsSuperperfect(G)` Inputs a group G and returns "true" if both the first and second integral homology of G is tri
`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
`SolutionsMatDestructive(M,B)` Inputs an $m\tilde{A}n$ matrix M and a $k\tilde{A}n$ matrix B over a field. It returns a k×m ma
`TestHap()` This runs a representative sample of HAP functions and checks to see that they produce the correct ou

Index