

Convex

**A GAP package for handling convex
objects.**

Version 2012.11.15

August 2012

Sebastian Gutsche

This manual is best viewed as an HTML document. An OFFLINE version should be included in the documentation subfolder of the package.

Sebastian Gutsche

Email: sebastian.gutsche@rwth-aachen.de

Homepage: <http://wwb.math.rwth-aachen.de/~gutsche>

Address: Lehrstuhl B für Mathematik, RWTH Aachen, Templergraben 64, 52056 Aachen, Germany

Copyright

© 2011-2012 by Sebastian Gutsche

This package may be distributed under the terms and conditions of the GNU Public License Version 2.

Acknowledgements

Contents

1	Introduction	4
1.1	What is the goal of the Convex package?	4
2	Installation of the Convex Package	5
3	Convex Objects	6
3.1	Convex Objects: Category and Representations	6
3.2	Convex objects: Properties	6
3.3	Convex objects: Attributes	6
3.4	Convex objects: Methods	7
4	Fan	8
4.1	Fan: Category and Representations	8
4.2	Fan: Properties	8
4.3	Fan: Attributes	9
4.4	Fan: Methods	9
4.5	Fan: Constructors	10
4.6	Fan: Examples	10
5	Cone	11
5.1	Cone: Category and Representations	11
5.2	Cone: Properties	11
5.3	Cone: Attributes	11
5.4	Cone: Methods	13
5.5	Cone: Constructors	13
5.6	Cone: Examples	13
6	Polytope	15
6.1	Polytope: Category and Representations	15
6.2	Polytope: Properties	15
6.3	Polytope: Attributes	16
6.4	Polytope: Methods	17
6.5	Polytope: Constructors	17
6.6	Polytope: Examples	17
	Index	19

Chapter 1

Introduction

1.1 What is the goal of the **Convex** package?

Convex provides structures and algorithms for convex geometry. It can handle convex, fans and polytopes. Not only the structures are provided, but also a collection of algorithms to handle those objects. Basically, it provides convex geometry to GAP. It is capable of communicating with the CAS polymake via the package `PolymakeInterface` and also provides several methods by itself.

Chapter 2

Installation of the **Convex** Package

To install this package just extract the package's archive file to the **GAP** pkg directory.

By default the **Convex** package is not automatically loaded by **GAP** when it is installed. You must load the package with

```
LoadPackage( "Convex" );
```

before its functions become available.

Please, send me an e-mail if you have any questions, remarks, suggestions, etc. concerning this package. Also, I would be pleased to hear about applications of this package and about any suggestions for new methods to add to the package.

Sebastian Gutsche

Chapter 3

Convex Objects

Convex objects are the main structure of `Convex`. All other structures, namely fans, cones, and polytopes are derived from this structure. So all methods of this structure also apply to the other data types.

3.1 Convex Objects: Category and Representations

3.1.1 `IsConvexObject`

▷ `IsConvexObject(M)` (Category)
Returns: `true` or `false`
The GAP category of convex objects, the main category of this package.

3.2 Convex objects: Properties

3.2.1 `IsFullDimensional`

▷ `IsFullDimensional(conv)` (property)
Returns: `true` or `false`
Checks if the combinatorial dimension of the convex object *conv* is the same as the dimension of the ambient space.

3.3 Convex objects: Attributes

3.3.1 `Dimension`

▷ `Dimension(conv)` (attribute)
Returns: an integer
Returns the combinatorial dimension of the convex object *conv*. This is the dimension of the smallest space *i* which *conv* can be embedded.

3.3.2 `AmbientSpaceDimension`

▷ `AmbientSpaceDimension(conv)` (attribute)
Returns: an integer

Returns the dimension of the ambient space of the object *conv*.

3.3.3 ContainingGrid

▷ `ContainingGrid(conv)` (attribute)

Returns: a homalg module

Returns the ambient space of the object *conv* as a homalg module.

3.4 Convex objects: Methods

3.4.1 DrawObject

▷ `DrawObject(conv)` (operation)

Returns: 0

Draws a nice picture of the object *conv*, if your computer supports Java. As a side effect, you might not be able to exit GAP anymore.

3.4.2 WeakPointerToExternalObject

▷ `WeakPointerToExternalObject(conv)` (operation)

Returns: a pointer

Returns a pointer to an external object which is the basis of *conv*. This method is not used any more.

Chapter 4

Fan

4.1 Fan: Category and Representations

4.1.1 IsFan

- ▷ `IsFan(M)` (Category)
Returns: true or false
The GAP category of a fan. Every fan is a convex object.
Remember: Every fan is a convex object.

4.2 Fan: Properties

4.2.1 IsComplete

- ▷ `IsComplete(fan)` (property)
Returns: true or false
Checks if the fan fan is complete, i. e. if it's support is the whole space.

4.2.2 IsPointed

- ▷ `IsPointed(fan)` (property)
Returns: true or false
Checks if the fan fan is pointed, which means that every cone it contains is strictly convex.

4.2.3 IsSmooth

- ▷ `IsSmooth(fan)` (property)
Returns: true or false
Checks if the fan fan is smooth, i. e. if every cone in the fan is smooth.

4.2.4 IsRegularFan

- ▷ `IsRegularFan(fan)` (property)
Returns: true or false
Checks if the fan fan is regular, i. e. if it is the normal fan of a polytope.

4.2.5 IsSimplicial (for a fan)

- ▷ `IsSimplicial(fan)` (property)
Returns: true or false
 Checks if the fan *fan* is simplicial, i. e. if every cone in the fan is simplicial.

4.2.6 HasConvexSupport

- ▷ `HasConvexSupport(fan)` (property)
Returns: true or false
 Checks if the fan *fan* is simplicial, i. e. if every cone in the fan is simplicial.

4.3 Fan: Attributes

4.3.1 Rays

- ▷ `Rays(fan)` (attribute)
Returns: a list
 Returns the rays of the fan *fan* as a list of cones.

4.3.2 RayGenerators

- ▷ `RayGenerators(fan)` (attribute)
Returns: a list
 Returns the generators rays of the fan *fan* as a list of list of integers.

4.3.3 RaysInMaximalCones

- ▷ `RaysInMaximalCones(fan)` (attribute)
Returns: a list
 Returns a list of lists, which represent an incidence matrix for the correspondence of the rays and the maximal cones of the fan *fan*. The *i*th list in the result represents the *i*th maximal cone of *fan*. In such a list, the *j*th entry is 1 if the *j*th ray is in the cone, 0 otherwise.

4.3.4 MaximalCones

- ▷ `MaximalCones(fan)` (attribute)
Returns: a list
 Returns the maximal cones of the fan *fan* as a list of cones.

4.4 Fan: Methods

4.4.1 * (for fans)

- ▷ `*(fan1, fan2)` (operation)
Returns: a fan
 Returns the product of the fans *fan1* and *fan2*.

4.5 Fan: Constructors

4.5.1 Fan (For Fans)

▷ `Fan(fan)` (operation)
Returns: a fan
 Copy constructor for fans. For completeness reasons.

4.5.2 Fan (For a list of rays and a list of cones)

▷ `Fan(rays, cones)` (operation)
Returns: a fan
 Constructs the fan out of the given *rays* and a list of *cones* given by a lists of numbers of rays.

4.6 Fan: Examples

4.6.1 Fan example

Example

```
gap> F := Fan( [[-1,5],[0,1],[1,0],[0,-1]], [[1,2],[2,3],[3,4],[4,1]] );
<A fan in |R^2>
gap> RayGenerators( F );
[[ -1, 5 ], [ 0, 1 ], [ 1, 0 ], [ 0, -1 ] ]
gap> RaysInMaximalCones( F );
[[ 1, 1, 0, 0 ], [ 0, 1, 1, 0 ], [ 0, 0, 1, 1 ], [ 1, 0, 0, 1 ] ]
gap> IsRegularFan( F );
true
gap> IsComplete( F );
true
gap> IsSmooth( F );
true
gap> F1 := MaximalCones( F )[ 1 ];
<A cone in |R^2>
gap> DualCone( F1 );
<A cone in |R^2>
gap> RayGenerators( F1 );
[[ -1, 5 ], [ 0, 1 ] ]
gap> F2 := StarSubdivisionOfIthMaximalCone( F, 1 );
<A fan in |R^2>
gap> IsSmooth( F2 );
true
gap> RayGenerators( F2 );
[[ -1, 5 ], [ -1, 6 ], [ 0, -1 ], [ 0, 1 ], [ 1, 0 ] ]
```

Chapter 5

Cone

5.1 Cone: Category and Representations

5.1.1 IsCone

▷ `IsCone(M)` (Category)
Returns: true or false
The GAP category of a cone.
Remember: Every cone is a convex object.

5.2 Cone: Properties

5.2.1 IsRay

▷ `IsRay($cone$)` (property)
Returns: true or false
Checks if the cone $cone$ is a ray, i.e. if it has only one ray generator.

5.3 Cone: Attributes

5.3.1 DualCone

▷ `DualCone($cone$)` (attribute)
Returns: a cone
Returns the dual cone of the cone $cone$.

5.3.2 HilbertBasis

▷ `HilbertBasis($cone$)` (attribute)
Returns: a list
Returns a Hilbert Basis of the cone $cone$.

5.3.3 RaysInFacets

▷ `RaysInFacets(cone)` (attribute)

Returns: a list

Returns an incidence matrix for the rays in the facets of the cone *cone*. The *i*th entry of the result corresponds to the *i*th facet, the *j*th entry of this is 1 if the *j*th ray is in the *i*th facet, 0 otherwise.

5.3.4 Facets

▷ `Facets(cone)` (attribute)

Returns: a list

Returns a list of the facets of the cone *cone* as homalg cones.

5.3.5 GridGeneratedByCone

▷ `GridGeneratedByCone(cone)` (attribute)

Returns: a homalg module

Returns the grid generated by the lattice points of the cone *cone* as a homalg module.

5.3.6 FactorGrid

▷ `FactorGrid(cone)` (attribute)

Returns: a homalg module

Returns the factor of the containing grid of the cone *cone* and the grid generated by *cone*.

5.3.7 GridGeneratedByOrthogonalCone

▷ `GridGeneratedByOrthogonalCone(cone)` (attribute)

Returns: a homalg module

Returns the grid generated by the lattice points of the orthogonal cone of the cone *cone*.

5.3.8 DefiningInequalities

▷ `DefiningInequalities(cone)` (attribute)

Returns: a list

Returns a list of the defining inequalities of the cone *cone*.

5.3.9 IsContainedInFan

▷ `IsContainedInFan(cone)` (attribute)

Returns: a fan

If the cone *cone* is constructed as part of a fan, this method returns the fan.

5.3.10 FactorGridMorphism

▷ `FactorGridMorphism(cone)` (attribute)

Returns: a morphism

Returns the morphism to the factor grid of the cone *cone*.

5.4 Cone: Methods

5.4.1 IntersectionOfCones

- ▷ `IntersectionOfCones(cone1, cone2)` (operation)
Returns: a cone
 If the cones *cone1* and *cone2* share a face, the method returns their intersection,

5.4.2 Contains

- ▷ `Contains(cone1, cone2)` (operation)
Returns: true or false
 Returns true if the cone *cone1* contains the cone *cone2*, false otherwise.

5.4.3 StarFan (for a cone)

- ▷ `StarFan(cone)` (operation)
Returns: a fan
 Returns the star fan of the cone *cone*, as described in cox, 3.2.7

5.4.4 StarFan (for a cone and a fan)

- ▷ `StarFan(cone, fan)` (operation)
Returns: a fan
 Returns the star fan of the fan *fan* along the cone *cone*, as described in cox, 3.2.7

5.4.5 StarSubdivisionOfIthMaximalCone

- ▷ `StarSubdivisionOfIthMaximalCone(fan, numb)` (operation)
Returns: a fan
 Returns the star subdivision of the fan *fan* on the *numb*th maximal cone as in cox, 3.3.13.

5.5 Cone: Constructors

5.5.1 Cone (for a ray list)

- ▷ `Cone(cone)` (operation)
Returns: a cone
 Returns a cone generated by the rays in *cone*.

5.6 Cone: Examples

5.6.1 Cone example

Example

```
gap> C := Cone([[1,2,3],[2,1,1],[1,0,0],[0,1,1]]);
<A cone in |R^3>
gap> Length( RayGenerators( C ) );
3
gap> IsSmooth( C );
```

```
true
gap> Length( HilbertBasis( C ) );
3
gap> IsSimplicial( C );
true
gap> DC := DualCone( C );
<A cone in |R^3>
gap> Length( HilbertBasis( DC ) );
3
```

Chapter 6

Polytope

6.1 Polytope: Category and Representations

6.1.1 IsPolytope

- ▷ `IsPolytope(M)` (Category)
Returns: true or false
The GAP category of a polytope. Every polytope is a convex object.
Remember: Every cone is a convex object.

6.2 Polytope: Properties

6.2.1 IsNotEmpty

- ▷ `IsNotEmpty($poly$)` (property)
Returns: true or false
Checks if the polytope $poly$ is not empty.

6.2.2 IsLatticePolytope

- ▷ `IsLatticePolytope($poly$)` (property)
Returns: true or false
Checks if the polytope $poly$ is a lattice polytope, i.e. all its vertices are lattice points.

6.2.3 IsVeryAmple

- ▷ `IsVeryAmple($poly$)` (property)
Returns: true or false
Checks if the polytope $poly$ is very ample.

6.2.4 IsNormalPolytope

- ▷ `IsNormalPolytope($poly$)` (property)
Returns: true or false
Checks if the polytope $poly$ is normal.

6.2.5 IsSimplicial (for a polytope)

- ▷ `IsSimplicial(poly)` (property)
Returns: true or false
 Checks if the polytope *poly* is simplicial.

6.2.6 IsSimplePolytope

- ▷ `IsSimplePolytope(poly)` (property)
Returns: true or false
 Checks if the polytope *poly* is simple.

6.3 Polytope: Attributes

6.3.1 Vertices

- ▷ `Vertices(poly)` (attribute)
Returns: a list
 Returns the vertices of the polytope *poly*. For reasons, the corresponding tester is `HasVerticesOfPolytopes`

6.3.2 LatticePoints

- ▷ `LatticePoints(poly)` (attribute)
Returns: a list
 Returns the lattice points of the polytope *poly*.

6.3.3 FacetInequalities

- ▷ `FacetInequalities(poly)` (attribute)
Returns: a list
 Returns the facet inequalities for the polytope *poly*.

6.3.4 VerticesInFacets

- ▷ `VerticesInFacets(poly)` (attribute)
Returns: a list
 Returns the incidence matrix of vertices and facets of the polytope *poly*.

6.3.5 AffineCone

- ▷ `AffineCone(poly)` (attribute)
Returns: a cone
 Returns the affine cone of the polytope *poly*.

6.3.6 NormalFan

- ▷ `NormalFan(poly)` (attribute)
Returns: a fan
 Returns the normal fan of the polytope *poly*.

6.3.7 RelativeInteriorLatticePoints

- ▷ `RelativeInteriorLatticePoints(poly)` (attribute)
Returns: a list
 Returns the lattice points in the relative interior of the polytope *poly*.

6.4 Polytope: Methods

6.4.1 * (for polytopes)

- ▷ `*(polytope1, polytope2)` (operation)
Returns: a polytope
 Returns the Cartesian product of the polytopes *polytope1* and *polytope2*.

6.4.2

- ▷ `#(polytope1, polytope2)` (operation)
Returns: a polytope
 Returns the Minkowski sum of the polytopes *polytope1* and *polytope2*.

6.5 Polytope: Constructors

6.5.1 Polytope (for lists of points)

- ▷ `Polytope(points)` (operation)
Returns: a polytope
 Returns a polytope that is the convex hull of the points *points*.

6.5.2 PolytopeByInequalities

- ▷ `PolytopeByInequalities(ineqs)` (operation)
Returns: a polytope
 Returns a polytope defined by the inequalities *ineqs*.

6.6 Polytope: Examples

6.6.1 Polytope example

Example

```
gap> P := Polytope( [ [ 2, 0 ], [ 0, 2 ], [ -1, -1 ] ] );
<A polytope in |R^2>
gap> IsVeryAmple( P );
true
```

```
gap> LatticePoints( P );  
[ [ -1, -1 ], [ 0, 0 ], [ 0, 1 ],  
  [ 0, 2 ], [ 1, 0 ], [ 1, 1 ], [ 2, 0 ] ]  
gap> NFP := NormalFan( P );  
<A complete fan in |R^2>  
gap> C1 := MaximalCones( NFP )[ 1 ];  
<A cone in |R^2>  
gap> RayGenerators( C1 );  
[ [ -1, -1 ], [ -1, 3 ] ]  
gap> IsRegularFan( NFP );  
true
```

Index

- #, [17](#)
- *
 - for fans, [9](#)
 - for polytopes, [17](#)
- Convex, [4](#)
- AffineCone, [16](#)
- AmbientSpaceDimension, [6](#)
- Cone
 - for a ray list, [13](#)
- ContainingGrid, [7](#)
- Contains, [13](#)
- DefiningInequalities, [12](#)
- Dimension, [6](#)
- DrawObject, [7](#)
- DualCone, [11](#)
- FacetInequalities, [16](#)
- Facets, [12](#)
- FactorGrid, [12](#)
- FactorGridMorphism, [12](#)
- Fan
 - For a list of rays and a list of cones, [10](#)
 - For Fans, [10](#)
- GridGeneratedByCone, [12](#)
- GridGeneratedByOrthogonalCone, [12](#)
- HasConvexSupport, [9](#)
- HilbertBasis, [11](#)
- IntersectionOfCones, [13](#)
- IsComplete, [8](#)
- IsCone, [11](#)
- IsContainedInFan, [12](#)
- IsConvexObject, [6](#)
- IsFan, [8](#)
- IsFullDimensional, [6](#)
- IsLatticePolytope, [15](#)
- IsNormalPolytope, [15](#)
- IsEmpty, [15](#)
- IsPointed, [8](#)
- IsPolytope, [15](#)
- IsRay, [11](#)
- IsRegularFan, [8](#)
- IsSimplePolytope, [16](#)
- IsSimplicial
 - for a fan, [9](#)
 - for a polytope, [16](#)
- IsSmooth, [8](#)
- IsVeryAmple, [15](#)
- LatticePoints, [16](#)
- MaximalCones, [9](#)
- NormalFan, [17](#)
- Polytope
 - for lists of points, [17](#)
- PolytopeByInequalities, [17](#)
- RayGenerators, [9](#)
- Rays, [9](#)
- RaysInFacets, [12](#)
- RaysInMaximalCones, [9](#)
- RelativeInteriorLatticePoints, [17](#)
- StarFan
 - for a cone, [13](#)
 - for a cone and a fan, [13](#)
- StarSubdivisionOfIthMaximalCone, [13](#)
- Vertices, [16](#)
- VerticesInFacets, [16](#)
- WeakPointerToExternalObject, [7](#)