NConvex

new version of the Package Convex

0.1

14/01/2016

Kamal Saleh

Kamal Saleh
Email: kamal.saleh@rwth-aachen.de
Homepage: Kamal.saleh@rwth-aachen.de
Address: Templergraben

Contents

Chapter 1

Cones

1.1 Creating cones

1.1.1 ConeByInequalities (for IsList)

▷ `ConeByInequalities(arg)` (operation)

Returns: a *Cone* Object

The function takes a list in which every entry represents an inequality and returns the cone defined by them.

1.1.2 ConeByEqualitiesAndInequalities (for IsList, IsList)

▷ `ConeByEqualitiesAndInequalities(arg)` (operation)

Returns: a *Cone* Object

The function takes two lists. The first list is the equalities and the second is the inequalities and returns the cone defined by them.

1.1.3 Cone (for IsList)

▷ `Cone(arg)` (operation)

Returns: a *Cone* Object

The function takes a list in which every entry represents a ray in the ambient vector space and returns the cone defined by them.

1.1.4 Cone (for IsCddPolyhedron)

▷ `Cone(cdd_cone)` (operation)

Returns: a *Cone* Object

This function takes a cone defined in *CddInterface* and converts it to a cone in *NConvex*

1.2 Attributes of Cones

1.2.1 DefiningInequalities (for IsCone)

▷ `DefiningInequalities(cone)` (attribute)

Returns: a *List*

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

1.2.2 EqualitiesOfCone (for IsCone)

▷ `EqualitiesOfCone(`*cone*`)` (attribute)
Returns: a *List*
Returns the list of the equalities in the defining inequalities of the cone *cone*.

1.2.3 DualCone (for IsCone)

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

1.2.4 Faces (for IsCone)

▷ `Faces(`*cone*`)` (attribute)
Returns: a list of cones
Returns the list of all faces of the cone *cone*.

1.2.5 Facets (for IsCone)

▷ `Facets(`*cone*`)` (attribute)
Returns: a list of cones
Returns the list of all faces of the cone *cone*.

1.2.6 RelativeInteriorRayGenerator (for IsCone)

▷ `RelativeInteriorRayGenerator(`*cone*`)` (attribute)
Returns: a point
Returns an interior point in the cone *cone*.

1.2.7 HilbertBasis (for IsCone)

▷ `HilbertBasis(`*cone*`)` (attribute)
Returns: a list
Returns the Hilbert basis of the cone *cone*

1.2.8 HilbertBasisOfDualCone (for IsCone)

▷ `HilbertBasisOfDualCone(`*cone*`)` (attribute)
Returns: a list
Returns the Hilbert basis of the dual cone of the cone *cone*

1.2.9 LinealitySpaceGenerators (for IsCone)

▷ `LinealitySpaceGenerators(`*cone*`)` (attribute)
Returns: a list
Returns a basis of the lineality space of the cone *cone*.

1.2.10 ExternalCddCone (for IsCone)

▷ `ExternalCddCone(`*cone*`)` (attribute)

Returns: a CddPolyhedron

Converts the cone to a CddPolyhedron. The functions of CddInterface can then be applied on this polyhedron.

1.3 Properties of Cones

1.3.1 IsRegularCone (for IsCone)

▷ `IsRegularCone(`*cone*`)` (property)

Returns: true or false

Returns if the cone *cone* is regular or not.

1.3.2 IsEmptyCone (for IsCone)

▷ `IsEmptyCone(`*cone*`)` (property)

Returns: true or false

Returns if the cone *cone* is empty or not.

1.3.3 IsRay (for IsCone)

▷ `IsRay(`*cone*`)` (property)

Returns: true or false

Returns if the cone *cone* is ray or not.

1.3.4 IsContainedInFan (for IsCone)

▷ `IsContainedInFan(`*cone*`)` (attribute)

Returns: true or false

Returns if the cone *cone* is contained in fan or not.

1.4 Operations on cones

1.4.1 FourierProjection (for IsCone, IsInt)

▷ `FourierProjection(`*cone, m*`)` (operation)

Returns: a cone

Returns the projectiopn of the cone on the space (O, $x_1, ..., x_{m-1}, x_{m+1}, ..., x_n$).

1.4.2 IntersectionOfCones (for IsCone, IsCone)

▷ `IntersectionOfCones(`*cone1, cone2*`)` (operation)

Returns: a cone

Returns the intersection of the cones *cone1* and *cone2*.

1.4.3 IntersectionOfConelist (for IsList)

▷ `IntersectionOfConelist(`*`[cone1, cone2, ...]`*`)` (operation)

Returns: a cone

Returns the intersection of all cones in the list *`[cone1, cone2, ...]`*.

1.4.4 Contains (for IsCone, IsCone)

▷ `Contains(`*`cone1, cone2`*`)` (operation)

Returns: a true or false

Returns if the cone *`cone1`* contains the cone *`cone2`*.

1.4.5 RayGeneratorContainedInCone (for IsList, IsCone)

▷ `RayGeneratorContainedInCone(`*`ray, cone`*`)` (operation)

Returns: true or false

Returns if the cone *`cone`* contains the ray *`ray`*.

Example

```
gap> P:= Cone( [ [ 2, 7 ], [ 0, 12 ], [ -2, 5 ] ] );
<A cone in |R^2>
gap> d:= DefiningInequalities( P );
[ [ -7, 2 ], [ 5, 2 ] ]
gap> Q:= ConeByInequalities( d );
<A cone in |R^2>
gap> P=Q;
true
gap> IsPointed( P );
true
gap> RayGenerators( P );
[ [ 2, 7 ], [ -2, 5 ] ]
gap> HilbertBasis( P );
[ [ -2, 5 ], [ -1, 3 ], [ 0, 1 ], [ 1, 4 ], [ 2, 7 ] ]
gap> HilbertBasis( Q );
[ [ -2, 5 ], [ -1, 3 ], [ 0, 1 ], [ 1, 4 ], [ 2, 7 ] ]
gap> P_dual:= DualCone( P );
<A cone in |R^2>
gap> RayGenerators( P_dual );
[ [ -7, 2 ], [ 5, 2 ] ]
gap> Dimension( P );
2
gap> Facets( P );
[ <A ray in |R^2>, <A ray in |R^2> ]
gap> List( last, RayGenerators );
[ [ [ 2, 7 ] ], [ [ -2, 5 ] ] ]
gap> faces := Faces( P );
[ <A cone in |R^2>, <A ray in |R^2>, <A ray in |R^2> ]
gap> RelativeInteriorRayGenerator( P );
[ -2, 29 ]
gap> LinealitySpaceGenerators( P );
[  ]
gap> IsRegularCone( P );
false
```

```
gap> IsEmptyCone( P );
false
gap> IsRay( P );
false
gap> proj_x1:= FourierProjection( P, 2 );
<A cone in |R^1>
gap> RayGenerators( proj_x1 );
[ [ 1 ], [ -1 ] ]
gap> DefiningInequalities( proj_x1 );
[ [ 0 ] ]
gap> R:= Cone( [ [ 4, 5 ], [ -2, 1 ] ] );
<A cone in |R^2>
gap> T:= IntersectionOfCones( P, R );
<A cone in |R^2>
gap> RayGenerators( T );
[ [ -2, 5 ], [ 2, 7 ] ]
gap> W:= Cone( [ [-3,-4 ] ] );
<A ray in |R^2>
gap> I:= IntersectionOfCones( P, W );
<A cone in |R^2>
gap> RayGenerators( I );
[  ]
gap> Contains( P, I );
true
gap> Contains( W, I );
true
gap> Contains( P, R );
false
gap> Contains( R, P );
true
gap> cdd_cone:= ExternalCddCone( P );
< Polyhedron given by its V-representation >
gap> Display( cdd_cone );
V-representation
begin
3 X 3  rational

   0   2   7
   0   0  12
   0  -2   5
end
gap> Cdd_Dimension( cdd_cone );
2
gap> H:= Cdd_H_Rep( cdd_cone );
< Polyhedron given by its H-representation >
gap> Display( H );
H-representation
begin
   2 X 3  rational

   0  -7   2
   0   5   2
end
```

```
gap> P:= Cone( [ [ 1, 1, -3 ], [ -1, -1, 3 ], [ 1, 2, 1 ], [ 2, 1, 2 ] ] );
< A cone in |R^3>
gap> IsPointed( P );
false
gap> Dimension( P );
3
gap> IsRegularCone( P );
false
gap> P;
< A cone in |R^3 of dimension 3 with 4 ray generators>
gap> RayGenerators( P );
[ [ 1, 1, -3 ], [ -1, -1, 3 ], [ 1, 2, 1 ], [ 2, 1, 2 ] ]
gap> d:= DefiningInequalities( P );
[ [ -5, 8, 1 ], [ 7, -4, 1 ] ]
gap> facets:= Facets( P );
[ <A cone in |R^3>, <A cone in |R^3> ]
gap> faces := Faces( P );
[ <A cone in |R^3>, <A cone in |R^3>, <A cone in |R^3>, <A cone in |R^3> ]
gap> FVector( P );
[ 1, 2, 1 ]
gap> List( faces, Dimension );
[ 3, 2, 1, 2 ]
gap> LatticePointsGenerators( P );
[ [ [ 0, 0, 0 ] ], [ [ 2, 1, 2 ], [ 1, 1, -2 ], [ 1, 2, 1 ] ], [ [ 1, 1, -3 ] ] ]
gap> DualCone( P );
< A cone in |R^3>
gap> RayGenerators( last );
[ [ -5, 8, 1 ], [ 7, -4, 1 ] ]
gap> Q_x1x3:= FourierProjection(P, 2 );
<A cone in |R^2>
gap> RayGenerators( Q_x1x3 );
[ [ 1, -3 ], [ -1, 3 ], [ 1, 1 ] ]
```

Chapter 2

NConvex automatic generated documentation

2.1 NConvex automatic generated documentation of global functions

2.1.1 NConvex_Example

▷ `NConvex_Example(`*arg*`)` (function)

Returns:
Insert documentation for you function here

Index