cohomCalgInterface

A package to communicate with the software cohomCalg

2019.09.10

10 September 2019

Martin Bies

Martin Bies
Email: martin.bies@alumni.uni-heidelberg.de
Homepage: https://www.ulb.ac.be/sciences/ptm/pmif/people.html
Address: Physique Théorique et Mathématique
Université Libre de Bruxelles
Campus Plaine - CP 231
Building NO - Level 6 - Office O.6.111
1050 Brussels
Belgium

Contents

1 Introduction **3**
1.1 What is the goal of the cohomCalg package? 3

2 Communication with cohomCalg **4**
2.1 Finding the cohomCalg binary 4
2.2 Turn toric variety and degree into a command string that we can pass to cohomCalg . 4
2.3 New attribute for toric varieties to store their monomial file 4

3 Functions supported by cohomCalg **5**
3.1 Functions to communicate with cohomCalg 5
3.2 Examples 6

Index **7**

Chapter 1

Introduction

1.1 What is the goal of the cohomCalg package?

cohomCalgInterface provides an interface, to communicate with the software cohomCalg via gap. Thereby, for example, line bundle cohomologies of toric varieties can be computed directly via gap.

Chapter 2

Communication with cohomCalg

2.1 Finding the cohomCalg binary

2.1.1 cohomCalgBinary

▷ `cohomCalgBinary(none)` (operation)

Returns: list [D, B]

This operation identifies the cohomCalg directory D and binary B

2.2 Turn toric variety and degree into a command string that we can pass to cohomCalg

2.2.1 cohomCalgCommandString (for IsToricVariety, IsList)

▷ `cohomCalgCommandString(vari, d)` (operation)

Returns: string

Given a toric variety vari and an element d of the class group, this operation prepares the command string that is handed over to cohomCalg

2.2.2 cohomCalgCommandString (for IsToricVariety)

▷ `cohomCalgCommandString(vari)` (operation)

Returns: string

This is a convenience method. Given a toric variety vari, it calls the previous method with the zero element of the class group.

2.3 New attribute for toric varieties to store their monomial file

2.3.1 MonomialFile (for IsToricVariety)

▷ `MonomialFile(vari)` (attribute)

Returns: a string

The name of the monomial file use for line bundle cohomology computations on this space.

Chapter 3

Functions supported by cohomCalg

3.1 Functions to communicate with cohomCalg

3.1.1 VanishingHiByCohomCalg (for IsToricVariety, IsList, IsInt)

▷ `VanishingHiByCohomCalg(vari, L, i)` (operation)

Returns: True or false

Given a toric variety vari and a list L = [D_2, ..., D_n] of degree in the class group, and thereby defining a direct sum of line bundles on vari. The third argument is a non-negative integer i. This function computes the i-th sheaf cohomology by use of cohomCalg and determines if it vanishes. The output is true or false.

3.1.2 AllHiByCohomCalg (for IsToricVariety, IsList)

▷ `AllHiByCohomCalg(vari, L)` (operation)

Returns: List

Given a toric variety vari and a list L = [D_2, ..., D_n] of degree in the class group, and thereby defining a direct sum of line bundles on vari, this function computes all of the sheaf cohomology dimensions of this vector bundle by use of cohomCalg. The output is a list of integers, each representing the corresponding sheaf cohomology dimension.

3.1.3 HiByCohomCalg (for IsToricVariety, IsInt, IsList)

▷ `HiByCohomCalg(vari, L, i)` (operation)

Returns: Integer

Given a toric variety vari and a list L = [D_2, ..., D_n] of degree in the class group, and thereby defining a direct sum of line bundles on vari. The third argument is a non-negative integer i. This function computes the i-th sheaf cohomology by use of cohomCalg.

3.1.4 ContributingDenominators (for IsToricVariety)

▷ `ContributingDenominators(vari)` (operation)

Returns: List

Given a toric variety vari, cohomCalg identifies line bundle cohomology as rationoms, i.e. fractions of monomials. This routine identifies all denominators of such rations, that appear as cohomologies of line bundles on the given toric variety.

3.2 Examples

Example

```
gap> P3 := ProjectiveSpace( 3 );
<A projective toric variety of dimension 3>
gap> coh1 := AllHiByCohomCalg( P3, [ [[3],1] ] );
[ 20, 0, 0, 0 ]
gap> P1 := ProjectiveSpace( 1 );
<A projective toric variety of dimension 1>
gap> P1xP1 := P1 * P1;
<A projective toric variety of dimension 2
which is a product of 2 toric varieties>
gap> coh2 := AllHiByCohomCalg( P1xP1, [ [[2,1],1] ] );
[ 6, 0, 0 ]
gap> coh3 := AllHiByCohomCalg( P1xP1, [ [[2,1],2] ] );
[ 12, 0, 0 ]
gap> coh4 := HiByCohomCalg( P1xP1, 0, [ [[2,1],2] ] );
12
gap> coh5 := HiByCohomCalg( P1xP1, 2, [ [[-2,-2],2] ] );
2
```

We also support the use of a monomial_file. Namely, cohomCalg computes combinatorial data on monomials in the Cox ring at each program run, unless this data is stored in a corresponding file. To use such a file, set the attribute "MonomialFile" to the desired name of this monomial file. Suppose for example, that we want to use the monomial file "Test.dat" for the space P3 above. Then we can use

Example

```
gap> SetMonomialFile( P3, "Test.dat" );
```

This file will be stored in the folder "cohomCalg" of this package.

Index

`AllHiByCohomCalg`
for IsToricVariety, IsList, 5

`cohomCalgBinary`, 4
`cohomCalgCommandString`
for IsToricVariety, 4
for IsToricVariety, IsList, 4
`ContributingDenominators`
for IsToricVariety, 5

`HiByCohomCalg`
for IsToricVariety, IsInt, IsList, 5

`MonomialFile`
for IsToricVariety, 4

`VanishingHiByCohomCalg`
for IsToricVariety, IsList, IsInt, 5