4ti2Interface

A link to 4ti2

2023.02-04

28 February 2023

Sebastian Gutsche

Sebastian Gutsche
Email: gutsche@mathematik.uni-siegen.de
Homepage: https://sebasguts.github.io
Address: Department Mathematik
Universität Siegen
Walter-Flex-Straße 3
57072 Siegen
Germany

Contents

Chapter 1

Introduction

1.1 What is the idea of 4ti2Interface

4ti2Interface is an GAP-Package that provides a link to the CAS 4ti2. It is not supposed to do any work by itself, but to provide the methods in 4ti2 to GAP. At the moment, it only capsules the groebner and hilbert method in 4ti2 but there are more to come. If you have any questions or suggestions, please feel free to contact me, or leave an issue on `https://github.com/homalg-project/4ti2Interface.git`.

Chapter 2

Installation

2.1 How to install this package

This package can only be used on a system that has 4ti2 installed. For more information about this please visit www.4ti2.de. For installing this package, first make sure you have 4ti2 installed. Copy it in your GAP pkg-directory. After this, the package can be loaded via LoadPackage("4ti2Interface");

Chapter 3

4ti2 functions

3.1 Groebner

These are wrappers of some use cases of 4ti2s groebner command.

3.1.1 4ti2Interface_groebner_matrix

▷ `4ti2Interface_groebner_matrix(`*`matrix[, ordering]`*`)` (function)

Returns: A list of vectors

This launches the 4ti2 groebner command with the argument as matrix input. The output will be the the Groebner basis of the binomial ideal generated by the left kernel of the input matrix. Note that this is different from 4ti2's convention which takes the right kernel. It returns the output of the groebner command as a list of lists. The second argument can be a vector to specify a monomial ordering, in the way that x^m > x^n if ordering*m > ordering*n

3.1.2 4ti2Interface_groebner_basis

▷ `4ti2Interface_groebner_basis(`*`basis[, ordering]`*`)` (function)

Returns: A list of vectors

This launches the 4ti2 groebner command with the argument as matrix input. The outpur will be the Groebner basis of the binomial ideal generated by the rows of the input matrix. It returns the output of the groebner command as a list of lists. The second argument is like before.

3.1.3 Defining ideal of toric variety

We want to compute the groebner basis of the ideal defining the affine toric variety associated to the cone generated by the inequalities [[7, -1], [0, 1]], i.e. a rational normal curve.

Example
```
gap> cone := [ [ 7, -1 ], [ 0, 1 ] ];
[ [ 7, -1 ], [ 0, 1 ] ]
gap> basis := 4ti2Interface_hilbert_inequalities( cone );;
gap> Sort( basis );
gap> basis;
[ [ 1, 0 ], [ 1, 1 ], [ 1, 2 ], [ 1, 3 ],
  [ 1, 4 ], [ 1, 5 ], [ 1, 6 ], [ 1, 7 ] ]
gap> groebner := 4ti2Interface_groebner_matrix( basis );;
gap> Sort( groebner );
gap> groebner;
[ [ -1, 0, 0, 1, 1, 0, 0, -1 ], [ -1, 0, 0, 2, 0, 0, -1, 0 ],
  [ -1, 0, 1, 0, 0, 1, 0, -1 ], [ -1, 0, 1, 0, 1, 0, -1, 0 ],
  [ -1, 0, 1, 1, 0, -1, 0, 0 ], [ -1, 0, 2, 0, -1, 0, 0, 0 ],
  [ -1, 1, 0, 0, 0, 0, 1, -1 ], [ -1, 1, 0, 0, 0, 1, -1, 0 ],
  [ -1, 1, 0, 0, 1, -1, 0, 0 ], [ -1, 1, 0, 1, -1, 0, 0, 0 ],
  [ -1, 1, 1, -1, 0, 0, 0, 0 ], [ -1, 2, -1, 0, 0, 0, 0, 0 ],
  [ 0, -1, 0, 0, 2, 0, 0, -1 ], [ 0, -1, 0, 1, 0, 1, 0, -1 ],
  [ 0, -1, 1, 0, 0, 0, 1, -1 ], [ 0, 0, -1, 0, 1, 1, 0, -1 ],
  [ 0, 0, -1, 1, 0, 0, 1, -1 ], [ 0, 0, 0, -1, 0, 2, 0, -1 ],
  [ 0, 0, 0, -1, 1, 0, 1, -1 ], [ 0, 0, 0, 0, -1, 1, 1, -1 ],
  [ 0, 0, 0, 0, 0, -1, 2, -1 ] ]
gap> Length( groebner );
21
```

3.2 Hilbert

These are wrappers of some use cases of 4ti2s hilbert command.

3.2.1 4ti2Interface_hilbert_inequalities

▷ `4ti2Interface_hilbert_inequalities(A)` (function)

▷ `4ti2Interface_hilbert_inequalities_in_positive_orthant(A)` (function)

Returns: a list of vectors

This function produces the hilbert basis of the cone C given by Ax >= 0 for all x in C. For the second function also x >= 0 is assumed.

3.2.2 4ti2Interface_hilbert_equalities_in_positive_orthant

▷ `4ti2Interface_hilbert_equalities_in_positive_orthant(A)` (function)

Returns: a list of vectors

This function produces the hilbert basis of the cone C given by the equations Ax = 0 in the positive orthant of the coordinate system.

3.2.3 4ti2Interface_hilbert_equalities_and_inequalities

▷ `4ti2Interface_hilbert_equalities_and_inequalities(A, B)` (function)

▷ `4ti2Interface_hilbert_equalities_and_inequalities_in_positive_orthant(A, B)` (function)

Returns: a list of vectors

This function produces the hilbert basis of the cone C given by the equations Ax = 0 and the inequations Bx >= 0. For the second function x>=0 is assumed.

3.2.4 Generators of semigroup

We want to compute the Hilbert basis of the cone obtained by intersecting the positive orthant with the hyperplane given by the equation below.

```
Example
gap> gens := [ 23, 25, 37, 49 ];
[ 23, 25, 37, 49 ]
gap> equation := [ Concatenation( gens, -gens ) ];
[ [ 23, 25, 37, 49, -23, -25, -37, -49 ] ]
gap> basis := 4ti2Interface_hilbert_equalities_in_positive_orthant( equation );;
gap> Length( basis );
436
```

3.2.5 Hilbert basis of dual cone

We want to compute the Hilbert basis of the cone which faces are represented by the inequalities below. This example is taken from the toric and the ToricVarieties package manual. In both packages it is very slow with the internal algorithms.

```
Example
gap> inequalities := [ [1,2,3,4], [0,1,0,7], [3,1,0,2], [0,0,1,0] ];
[ [ 1, 2, 3, 4 ], [ 0, 1, 0, 7 ], [ 3, 1, 0, 2 ], [ 0, 0, 1, 0 ] ]
gap> basis := 4ti2Interface_hilbert_inequalities( inequalities );;
gap> Length( basis );
29
```

3.3 ZSolve

3.3.1 4ti2Interface_zsolve_equalities_and_inequalities

▷ `4ti2Interface_zsolve_equalities_and_inequalities(`*eqs*`,` *eqs_rhs*`,` *ineqs*`,` *ineqs_rhs*`[,` *signs*`])` (function)

▷ `4ti2Interface_zsolve_equalities_and_inequalities_in_positive_orthant(`*eqs*`,` *eqs_rhs*`,` *ineqs*`,` *ineqs_rhs*`)` (function)

Returns: a list of three matrices

This function produces a basis of the system *eqs* = *eqs_rhs* and *ineqs* >= *ineqs_rhs*. It outputs a list containing three matrices. The first one is a list of points in a polytope, the second is the hilbert basis of a cone. The set of solutions is then the minkowski sum of the polytope generated by the points in the first list and the cone generated by the hilbert basis in the second matrix. The third one is the free part of the solution polyhedron. The optional argument *signs* must be a list of zeros and ones which length is the number of variables. If the ith entry is one, the ith variable must be >= 0. If the entry is 0, the number is arbitraty. Default is all zero. It is also possible to set the option precision to 32, 64 or gmp. The default, if no option is given, 32 is used. Please note that a higher precision leads to slower computation. For the second function xi >= 0 for all variables is assumed.

3.4 Graver

3.4.1 4ti2Interface_graver_equalities

▷ `4ti2Interface_graver_equalities(`*eqs*`[,` *signs*`])` (function)

▷ `4ti2Interface_graver_equalities_in_positive_orthant(`*eqs*`)` (function)

Returns: a matrix

This calls the function graver with the equalities `eqs` = 0. It outputs one list containing the graver basis of the system. the optional argument `signs` is used like in zsolve. The second command assumes $x_i \geq 0$.

Chapter 4

Tool functions

4.1 Read and write matrix

4.1.1 4ti2Interface_Read_Matrix_From_File

▷ `4ti2Interface_Read_Matrix_From_File(`*`arg`*`)` (function)

Returns: a list of vectors

The argument must be a string, representing a filename of a matrix to read. Numbers must be seperated by whitespace, and the first two numbers must be the number of rows and columns. The function then returns the matrix as list of lists.

4.1.2 4ti2Interface_Write_Matrix_To_File

▷ `4ti2Interface_Write_Matrix_To_File(`*`arg`*`)` (function)

Returns: nothing

First argument must be a matrix, i.e. a list of list of integers. Second argument has to be a filename. The method stores the matrix in this file, seperated by whitespace, line by line. The content of the file, if there is any, will be deleted.

4.1.3 4ti2Interface_Cut_Vector

▷ `4ti2Interface_Cut_Vector(`*`vec, d`*`)` (function)

Returns: a matrix

Takes the vector *`vec`* and produces a matrix with *`d`* columns out of the entries of the vector.

Index