SpasmInterface

A package to communicate with the software Spasm

2021.11.17

17 November 2021

Martin Bies

Martin Bies
Email: martin.bies@alumni.uni-heidelberg.de
Homepage: https://martinbies.github.io/
Address: Department of Mathematics
University of Pennsylvania
David Rittenhouse Laboratory
209 S 33rd St
Philadelphia
PA 19104

Copyright

Contents

Chapter 1

Introduction

1.1 What is the goal of the SpasmInterface package?

SpasmInterface provides an interface, to communicate with the software Spasm via gap. Thereby, for example, kernel of sparse matrices can be computed directly via gap.

Chapter 2

Interface to Spasm

2.1 Finding the SpasmDirectory

2.1.1 FindSpasmDirectory

▷ `FindSpasmDirectory(`*`none`*`)` (operation)

Returns: the corresponding directory

This operation identifies the location of the spasm programs.

2.2 Executing Spasm

2.2.1 ExecuteSpasm (for IsDirectory, IsString, IsList, IsList, IsList)

▷ `ExecuteSpasm(`*`A, Directory, a, string, and, two, lists`*`)` (operation)

Returns: the corresponding quantity as computed by Spasm as a string

This operation executes Spasm with four pieces of input information. The first is the directory of Spasm, the second the name of the binary that is to be executed, the third is the input required by this binary. The fourth argument is a list of options supported by Spasm and the fifth the list of values for these arguments.

Chapter 3

Functionality of Spasm

3.1 Computation of syzygies

3.1.1 SyzygiesOfRowsBySpasm (for IsSMSSparseMatrix)

▷ SyzygiesOfRowsBySpasm(*SMSSparseMatrix, M*) (operation)

Returns: SMSSparseMatrix

Compute left kernel of SMSSparseMatrix M by Spasm. By default we compute it over the finite field of order 42013. As second argument, an integer can be provided to overwrite this default.

3.1.2 SyzygiesOfRowsBySpasm (for IsSMSSparseMatrix, IsInt)

▷ SyzygiesOfRowsBySpasm(*arg1, arg2*) (operation)

3.1.3 SyzygiesOfColumnsBySpasm (for IsSMSSparseMatrix)

▷ SyzygiesOfColumnsBySpasm(*SMSSparseMatrix, M*) (operation)

Returns: SMSSparseMatrix

Compute right kernel of SMSSparseMatrix M by Spasm. By default we compute it over the finite field of order 42013. As second argument, an integer can be provided to overwrite this default.

3.1.4 SyzygiesOfColumnsBySpasm (for IsSMSSparseMatrix, IsInt)

▷ SyzygiesOfColumnsBySpasm(*arg1, arg2*) (operation)

3.1.5 RowSyzygiesGeneratorsBySpasm (for IsSMSSparseMatrix, IsSMSSparseMatrix)

▷ RowSyzygiesGeneratorsBySpasm(*SparseMatrices, M1, M2*) (operation)

Returns: SMSSparseMatrix

This function computes the RowSyzygyGenerators of two SMSSparseMatrices M1, M2. By default we compute it over the finite field of order 42013. As second argument, an integer can be provided to overwrite this default.

3.1.6 RowSyzygiesGeneratorsBySpasm (for IsSMSSparseMatrix, IsSMSSparseMatrix, IsInt)

▷ RowSyzygiesGeneratorsBySpasm(*arg1, arg2, arg3*) (operation)

3.1.7 ColumnSyzygiesGeneratorsBySpasm (for IsSMSSparseMatrix, IsSMSSparseMatrix)

▷ ColumnSyzygiesGeneratorsBySpasm(*SparseMatrices, M1, M2*) (operation)

Returns: SMSSparseMatrix

This function computes the ColumnSyzygyGenerators of two SMSSparseMatrices M1, M2. By default we compute it over the finite field of order 42013. As second argument, an integer can be provided to overwrite this default.

3.1.8 ColumnSyzygiesGeneratorsBySpasm (for IsSMSSparseMatrix, IsSMSSparseMatrix, IsInt)

▷ ColumnSyzygiesGeneratorsBySpasm(*arg1, arg2, arg3*) (operation)

3.2 Computation of rank

3.2.1 RankGPLUBySpasm (for IsSMSSparseMatrix)

▷ RankGPLUBySpasm(*SMSSparseMatrix, M*) (operation)

Returns: Integer

Compute the rank of an SMSSparseMatrix M by Spasm by GPLU. By default we compute it over the finite field of order 42013. As second argument, an integer can be provided to overwrite this default.

3.2.2 RankGPLUBySpasm (for IsSMSSparseMatrix, IsInt)

▷ RankGPLUBySpasm(*arg1, arg2*) (operation)

3.2.3 RankDenseBySpasm (for IsSMSSparseMatrix)

▷ RankDenseBySpasm(*SMSSparseMatrix, M*) (operation)

Returns: Integer

Compute the rank of an SMSSparseMatrix M by Spasm. This uses an algorithm designed for handeling dense matrices, but is here applied to a sparse matrix nonetheless. By default we compute it over the finite field of order 42013. As second argument, an integer can be provided to overwrite this default.

3.2.4 RankDenseBySpasm (for IsSMSSparseMatrix, IsInt)

▷ RankDenseBySpasm(*arg1, arg2*) (operation)

3.2.5 RankHybridBySpasm (for IsSMSSparseMatrix)

▷ RankHybridBySpasm(*SMSSparseMatrix, M*) (operation)

Returns: Integer

Compute the rank of an SMSSparseMatrix M by Spasm. This uses the hybrid strategy described in [PASCO'17]. By default we compute it over the finite field of order 42013. As second argument, an integer can be provided to overwrite this default.

3.2.6 RankHybridBySpasm (for IsSMSSparseMatrix, IsInt)

▷ RankHybridBySpasm(*arg1, arg2*) (operation)

3.3 Examples

We can compute syzygies of rows and columns as follows:

Example

```
gap> entries1 := [ [ 1, 1, 1 ], [ 2, 1, -1 ], [ 3, 2, 1 ] ];;
gap> m1 := SMSSparseMatrix( 3, 2, entries1 );;
gap> k1 := SyzygiesOfRowsBySpasm( m1 );;
gap> SyzygiesOfColumnsBySpasm( m1 );;
gap> NumberOfRows( k1 );
1
gap> NumberOfColumns( k1 );
3
```

Here is another example.

Example

```
gap> entries2 := [ [ 1, 2, 1 ], [ 2, 1, 1 ], [ 2, 2, 1 ], [ 3, 1, -1 ], [ 3, 2, 1 ] ];;
gap> m2 := SMSSparseMatrix( 3, 2, entries2 );;
gap> NumberOfRows( m2 );
3
gap> NumberOfColumns( m2 );
2
gap> s2 := TurnIntoSMSString( m2 );;
gap> k2 := SyzygiesOfRowsBySpasm( m2 );;
gap> SyzygiesOfColumnsBySpasm( m2 );;
gap> NumberOfRows( k2 );
1
gap> NumberOfColumns( k2 );
3
```

Given two sparse matrices, we can stack them in that we take the collection formed from their union of rows and interpret the result as a new sparse matrix. This is also used to compute the relative syzygies of a matrix $m1$ with respect to a second matrix $m2$. We demonstrate this with the above two matrices:

Example

```
gap> m3 := UnionOfRowsOp( m1, m2 );;
gap> NumberOfRows( m3 );
6
gap> NumberOfColumns( m3 );
2
gap> m4 := RowSyzygiesGeneratorsBySpasm( m1, m2 );
<A 4x3 sparse matrix in SMS-format>
gap> ColumnSyzygiesGeneratorsBySpasm( m1, m2 );
<A 2x1 sparse matrix in SMS-format>
gap> NumberOfRows( m4 );
4
gap> NumberOfColumns( m4 );
3
```

Spasm also supports various algorithms for computing ranks of sparse matrices. Here are some examples

Example

```
gap> RankGPLUBySpasm( m3 );
2
gap> RankDenseBySpasm( m3 );
2
gap> RankHybridBySpasm( m3 );
2
```

Index