

Linear Algebra For- CAP

**Category of Matrices over a Field for
CAP**

2024.06-01

10 June 2024

Sebastian Gutsche

Sebastian Posur

Fabian Zickgraf

Sebastian Gutsche

Email: gutsche@mathematik.uni-siegen.de

Homepage: <https://sebasguts.github.io/>

Address: Department Mathematik

Universität Siegen

Walter-Flex-Straße 3

57068 Siegen

Germany

Sebastian Posur

Email: sebastian.posur@uni-siegen.de

Homepage: <https://sebastianpos.github.io>

Address: Department Mathematik

Universität Siegen

Walter-Flex-Straße 3

57068 Siegen

Germany

Fabian Zickgraf

Email: fabian.zickgraf@uni-siegen.de

Homepage: <https://github.com/zickgraf/>

Address: Walter-Flex-Str. 3

57068 Siegen

Germany

Contents

1	Category of Matrices	3
1.1	Constructors	3
1.2	Attributes	4
1.3	GAP Categories	4
2	Examples and Tests	6
2.1	Basic Commands	6
2.2	Split epi summand	11
2.3	Kernel	11
2.4	FiberProduct	13
	Index	14

Chapter 1

Category of Matrices

1.1 Constructors

1.1.1 MatrixCategory (for IsFieldForHomalg)

▷ `MatrixCategory(F)` (attribute)

Returns: a category

The argument is a homalg field F . The output is the matrix category over F . Objects in this category are non-negative integers. Morphisms from a non-negative integer m to a non-negative integer n are given by $m \times n$ matrices.

1.1.2 MatrixCategoryAsCategoryOfRows (for IsFieldForHomalg)

▷ `MatrixCategoryAsCategoryOfRows(F)` (operation)

Returns: a category

The argument is a homalg field F . The output is the matrix category over F , constructed internally as a wrapper category of the `CategoryOfRows` of F . Only available if the package `FreydCategoriesForCAP` is available.

1.1.3 VectorSpaceMorphism (for IsVectorSpaceObject, IsHomalgMatrix, IsVectorSpaceObject)

▷ `VectorSpaceMorphism(S, M, R)` (operation)

Returns: a morphism in $\text{Hom}(S, R)$

The arguments are an object S in the category of matrices over a homalg field F , a homalg matrix M over F , and another object R in the category of matrices over F . The output is the morphism $S \rightarrow R$ in the category of matrices over F whose underlying matrix is given by M .

1.1.4 VectorSpaceObject (for IsInt, IsFieldForHomalg)

▷ `VectorSpaceObject(d, F)` (operation)

Returns: an object

The arguments are a non-negative integer d and a homalg field F . The output is an object in the category of matrices over F of dimension d . This function delegates to `MatrixCategoryObject`.

1.1.5 MatrixCategoryObject (for IsMatrixCategory, IsInt)

▷ `MatrixCategoryObject(cat, d)` (operation)

Returns: an object

The arguments are a matrix category cat over a field and a non-negative integer d . The output is an object in cat of dimension d .

1.2 Attributes

1.2.1 UnderlyingFieldForHomalg (for IsVectorSpaceMorphism)

▷ `UnderlyingFieldForHomalg(alpha)` (attribute)

Returns: a homalg field

The argument is a morphism α in the matrix category over a homalg field F . The output is the field F .

1.2.2 UnderlyingMatrix (for IsVectorSpaceMorphism)

▷ `UnderlyingMatrix(alpha)` (attribute)

Returns: a homalg matrix

The argument is a morphism α in a matrix category. The output is its underlying matrix M .

1.2.3 UnderlyingFieldForHomalg (for IsVectorSpaceObject)

▷ `UnderlyingFieldForHomalg(A)` (attribute)

Returns: a homalg field

The argument is an object A in the matrix category over a homalg field F . The output is the field F .

1.2.4 Dimension (for IsVectorSpaceObject)

▷ `Dimension(A)` (attribute)

Returns: a non-negative integer

The argument is an object A in a matrix category. The output is the dimension of A .

1.3 GAP Categories

1.3.1 IsVectorSpaceMorphism (for IsCapCategoryMorphism and IsCellOfSkeletal-Category)

▷ `IsVectorSpaceMorphism(object)` (filter)

Returns: true or false

The GAP category of morphisms in the category of matrices of a field F .

1.3.2 IsVectorSpaceObject (for IsCapCategoryObject and IsCellOfSkeletalCategory)

▷ `IsVectorSpaceObject(object)` (filter)

Returns: true or false

The GAP category of objects in the category of matrices of a field F .

Chapter 2

Examples and Tests

2.1 Basic Commands

Example

```
gap> Q := HomalgFieldOfRationals();;
gap> a := VectorSpaceObject( 3, Q );
<A vector space object over Q of dimension 3>
gap> HasIsProjective( a ) and IsProjective( a );
true
gap> vec := MatrixCategory( Q );;
gap> ap := 3/vec;;
gap> IsEqualForObjects( a, ap );
true
gap> b := VectorSpaceObject( 4, Q );
<A vector space object over Q of dimension 4>
gap> homalg_matrix := HomalgMatrix( [ [ 1, 0, 0, 0 ],
>                                     [ 0, 1, 0, -1 ],
>                                     [ -1, 0, 2, 1 ] ], 3, 4, Q );
<A 3 x 4 matrix over an internal ring>
gap> alpha := VectorSpaceMorphism( a, homalg_matrix, b );
<A morphism in Category of matrices over Q>
gap> Display( alpha );
[ [ 1, 0, 0, 0 ],
  [ 0, 1, 0, -1 ],
  [ -1, 0, 2, 1 ] ]

A morphism in Category of matrices over Q
gap> alphap := homalg_matrix/vec;;
gap> IsCongruentForMorphisms( alpha, alphap );
true
gap> homalg_matrix := HomalgMatrix( [ [ 1, 1, 0, 0 ],
>                                     [ 0, 1, 0, -1 ],
>                                     [ -1, 0, 2, 1 ] ], 3, 4, Q );
<A 3 x 4 matrix over an internal ring>
gap> beta := VectorSpaceMorphism( a, homalg_matrix, b );
<A morphism in Category of matrices over Q>
gap> CokernelObject( alpha );
<A vector space object over Q of dimension 1>
gap> c := CokernelProjection( alpha );;
```

```
#@if ValueOption( "no_precompiled_code" ) <> true
```

```
gap> Display( c );
```

```
[ [ 0 ],
  [ 1 ],
  [ -1/2 ],
  [ 1 ] ]
```

```
A split epimorphism in Category of matrices over Q
```

```
#@fi
```

```
gap> gamma := UniversalMorphismIntoDirectSum( [ c, c ] );;
```

```
gap> Display( gamma );
```

```
[ [ 0, 0 ],
  [ 1, 1 ],
  [ -1/2, -1/2 ],
  [ 1, 1 ] ]
```

```
A morphism in Category of matrices over Q
```

```
gap> colift := CokernelColift( alpha, gamma );;
```

```
gap> IsEqualForMorphisms( PreCompose( c, colift ), gamma );
```

```
true
```

```
gap> FiberProduct( alpha, beta );
```

```
<A vector space object over Q of dimension 2>
```

```
gap> F := FiberProduct( alpha, beta );
```

```
<A vector space object over Q of dimension 2>
```

```
gap> p1 := ProjectionInFactorOfFiberProduct( [ alpha, beta ], 1 );
```

```
<A morphism in Category of matrices over Q>
```

```
gap> Display( PreCompose( p1, alpha ) );
```

```
[ [ 0, 1, 0, -1 ],
  [ -1, 0, 2, 1 ] ]
```

```
A morphism in Category of matrices over Q
```

```
gap> Pushout( alpha, beta );
```

```
<A vector space object over Q of dimension 5>
```

```
gap> i1 := InjectionOfCofactorOfPushout( [ alpha, beta ], 1 );
```

```
<A morphism in Category of matrices over Q>
```

```
gap> i2 := InjectionOfCofactorOfPushout( [ alpha, beta ], 2 );
```

```
<A morphism in Category of matrices over Q>
```

```
gap> u := UniversalMorphismFromDirectSum( [ b, b ], [ i1, i2 ] );
```

```
<A morphism in Category of matrices over Q>
```

```
gap> Display( u );
```

```
[ [ 0, 1, 1, 0, 0 ],
  [ 1, 0, 1, 0, -1 ],
  [ -1/2, 0, 1/2, 1, 1/2 ],
  [ 1, 0, 0, 0, 0 ],
  [ 0, 1, 0, 0, 0 ],
  [ 0, 0, 1, 0, 0 ],
  [ 0, 0, 0, 1, 0 ],
  [ 0, 0, 0, 0, 1 ] ]
```

```
A morphism in Category of matrices over Q
```

```
gap> KernelObjectFunctorial( u, IdentityMorphism( Source( u ) ), u ) = IdentityMorphism( VectorSpace( Source( u ) ) );
```

```
true
```



```

gap> IsZero( CokernelObjectFunctorial( u, IdentityMorphism( Range( u ) ), u ) );
true
gap> DirectProductFunctorial( [ u, u ] ) = DirectSumFunctorial( [ u, u ] );
true
gap> CoproductFunctorial( [ u, u ] ) = DirectSumFunctorial( [ u, u ] );
true
gap> IsOne( FiberProductFunctorial( [ u, u ], [ IdentityMorphism( Source( u ) ), IdentityMorphism( Range( u ) ) ] ), IdentityMorphism( Source( u ) ) );
true
gap> IsOne( PushoutFunctorial( [ u, u ], [ IdentityMorphism( Range( u ) ), IdentityMorphism( Range( u ) ) ] ), IdentityMorphism( Range( u ) ) );
true
gap> IsCongruentForMorphisms( (1/2) * alpha, alpha * (1/2) );
true
gap> Dimension( HomomorphismStructureOnObjects( a, b ) ) = Dimension( a ) * Dimension( b );
true
gap> IsCongruentForMorphisms(
>   PreCompose( [ u, DualOnMorphisms( i1 ), DualOnMorphisms( alpha ) ] ),
>   InterpretMorphismFromDistinguishedObjectToHomomorphismStructureAsMorphism( Source( u ), Source( u ) ),
>   PreCompose(
>     InterpretMorphismAsMorphismFromDistinguishedObjectToHomomorphismStructure( DualOnMorphisms( i1 ), DualOnMorphisms( alpha ) ),
>     HomomorphismStructureOnMorphisms( u, DualOnMorphisms( alpha ) )
>   )
> );
true
gap> alpha_op := Opposite( alpha );
<A morphism in Opposite( Category of matrices over Q )>
gap> basis := BasisOfExternalHom( Source( alpha_op ), Range( alpha_op ) );
gap> coeffs := CoefficientsOfMorphism( alpha_op );
[ 1, 0, 0, 0, 0, 1, 0, -1, -1, 0, 2, 1 ]
gap> IsEqualForMorphisms( alpha_op, coeffs * basis );
true
gap> vec := CapCategory( alpha );
gap> t := TensorUnit( vec );
gap> z := ZeroObject( vec );
gap> IsCongruentForMorphisms(
>   ZeroObjectFunctorial( vec ),
>   InterpretMorphismFromDistinguishedObjectToHomomorphismStructureAsMorphism( z, z, ZeroMorphism( z ) )
> );
true
gap> IsCongruentForMorphisms(
>   ZeroObjectFunctorial( vec ),
>   InterpretMorphismFromDistinguishedObjectToHomomorphismStructureAsMorphism(
>     z, z,
>     InterpretMorphismAsMorphismFromDistinguishedObjectToHomomorphismStructure( ZeroObjectFunctorial( vec ), ZeroMorphism( z ) )
>   )
> );
true
gap> right_side := PreCompose( [ i1, DualOnMorphisms( u ), u ] );
gap> x := SolveLinearSystemInAbCategory( [ [ i1 ] ], [ [ u ] ], [ right_side ] )[1];
gap> IsCongruentForMorphisms( PreCompose( [ i1, x, u ] ), right_side );
true
gap> a_otimes_b := TensorProductOnObjects( a, b );

```

```

<A vector space object over Q of dimension 12>
gap> hom_ab := InternalHomOnObjects( a, b );
<A vector space object over Q of dimension 12>
gap> cohom_ab := InternalCoHomOnObjects( a, b );
<A vector space object over Q of dimension 12>
gap> hom_ab = cohom_ab;
true
gap> 1_ab := VectorSpaceMorphism(
>      a_otimes_b,
>      HomalgIdentityMatrix( Dimension( a_otimes_b ), Q ),
>      a_otimes_b
>      );
<A morphism in Category of matrices over Q>
gap> 1_hom_ab := VectorSpaceMorphism(
>      hom_ab,
>      HomalgIdentityMatrix( Dimension( hom_ab ), Q ),
>      hom_ab
>      );
<A morphism in Category of matrices over Q>
gap> 1_cohom_ab := VectorSpaceMorphism(
>      cohom_ab,
>      HomalgIdentityMatrix( Dimension( cohom_ab ), Q ),
>      cohom_ab
>      );
<A morphism in Category of matrices over Q>
gap> ev_ab := ClosedMonoidalLeftEvaluationMorphism( a, b );
<A morphism in Category of matrices over Q>
gap> coev_ab := ClosedMonoidalLeftCoevaluationMorphism( a, b );
<A morphism in Category of matrices over Q>
gap> coev_ba := ClosedMonoidalLeftCoevaluationMorphism( b, a );
<A morphism in Category of matrices over Q>
gap> cocl_ev_ab := CoclosedMonoidalLeftEvaluationMorphism( a, b );
<A morphism in Category of matrices over Q>
gap> cocl_ev_ba := CoclosedMonoidalLeftEvaluationMorphism( b, a );
<A morphism in Category of matrices over Q>
gap> cocl_coev_ab := CoclosedMonoidalLeftCoevaluationMorphism( a, b );
<A morphism in Category of matrices over Q>
gap> cocl_coev_ba := CoclosedMonoidalLeftCoevaluationMorphism( b, a );
<A morphism in Category of matrices over Q>
gap> UnderlyingMatrix( ev_ab ) = TransposedMatrix( UnderlyingMatrix( cocl_ev_ab ) );
true
gap> UnderlyingMatrix( coev_ab ) = TransposedMatrix( UnderlyingMatrix( cocl_coev_ab ) );
true
gap> UnderlyingMatrix( coev_ba ) = TransposedMatrix( UnderlyingMatrix( cocl_coev_ba ) );
true
gap> tensor_hom_adj_1_hom_ab := InternalHomToTensorProductLeftAdjunctMorphism( a, b, 1_hom_ab );
<A morphism in Category of matrices over Q>
gap> cohom_tensor_adj_1_cohom_ab := InternalCoHomToTensorProductLeftAdjunctMorphism( a, b, 1_cohom_ab );
<A morphism in Category of matrices over Q>
gap> tensor_hom_adj_1_ab := TensorProductToInternalHomLeftAdjunctMorphism( a, b, 1_ab );
<A morphism in Category of matrices over Q>
gap> cohom_tensor_adj_1_ab := TensorProductToInternalCoHomLeftAdjunctMorphism( a, b, 1_ab );

```

```

<A morphism in Category of matrices over Q>
gap> ev_ab = tensor_hom_adj_1_hom_ab;
true
gap> cocl_ev_ba = cohom_tensor_adj_1_cohom_ab;
true
gap> coev_ba = tensor_hom_adj_1_ab;
true
gap> cocl_coev_ba = cohom_tensor_adj_1_ab;
true
gap> c := VectorSpaceObject(2,Q);
<A vector space object over Q of dimension 2>
gap> d := VectorSpaceObject(1,Q);
<A vector space object over Q of dimension 1>
gap> pre_compose := MonoidalPreComposeMorphism( a, b, c );
<A morphism in Category of matrices over Q>
gap> post_compose := MonoidalPostComposeMorphism( a, b, c );
<A morphism in Category of matrices over Q>
gap> pre_cocompose := MonoidalPreCoComposeMorphism( c, b, a );
<A morphism in Category of matrices over Q>
gap> post_cocompose := MonoidalPostCoComposeMorphism( c, b, a );
<A morphism in Category of matrices over Q>
gap> UnderlyingMatrix( pre_compose ) = TransposedMatrix( UnderlyingMatrix( pre_cocompose ) );
true
gap> UnderlyingMatrix( post_compose ) = TransposedMatrix( UnderlyingMatrix( post_cocompose ) );
true
gap> tp_hom_comp := TensorProductInternalHomCompatibilityMorphism( [ a, b, c, d ] );
<A morphism in Category of matrices over Q>
gap> cohom_tp_comp := InternalCoHomTensorProductCompatibilityMorphism( [ b, d, a, c ] );
<A morphism in Category of matrices over Q>
gap> UnderlyingMatrix( tp_hom_comp ) = TransposedMatrix( UnderlyingMatrix( cohom_tp_comp ) );
true
gap> lambda := LambdaIntroduction( alpha );
<A morphism in Category of matrices over Q>
gap> lambda_elim := LambdaElimination( a, b, lambda );
<A morphism in Category of matrices over Q>
gap> alpha = lambda_elim;
true
gap> alpha_op := VectorSpaceMorphism( b, TransposedMatrix( UnderlyingMatrix( alpha ) ), a );
<A morphism in Category of matrices over Q>
gap> colambda := CoLambdaIntroduction( alpha_op );
<A morphism in Category of matrices over Q>
gap> colambda_elim := CoLambdaElimination( b, a, colambda );
<A morphism in Category of matrices over Q>
gap> alpha_op = colambda_elim;
true
gap> UnderlyingMatrix( lambda ) = TransposedMatrix( UnderlyingMatrix( colambda ) );
true
gap> delta := PreCompose( colambda, lambda );
<A morphism in Category of matrices over Q>
gap> Display( TraceMap( delta ) );
[ [ 9 ] ]

```

```

A morphism in Category of matrices over Q
gap> Display( CoTraceMap( delta ) );
[ [ 9 ] ]

A morphism in Category of matrices over Q
gap> TraceMap( delta ) = CoTraceMap( delta );
true
gap> RankMorphism( a ) = CoRankMorphism( a );
true

```

2.2 Split epi summand

Example

```

gap> Q := HomalgFieldOfRationals();;
gap> a := VectorSpaceObject( 3, Q );;
gap> b := VectorSpaceObject( 4, Q );;
gap> homalg_matrix := HomalgMatrix( [ [ 1, 0, 0, 0 ],
>                                     [ 0, 1, 0, -1 ],
>                                     [ -1, 0, 2, 1 ] ], 3, 4, Q );;
gap> alpha := VectorSpaceMorphism( a, homalg_matrix, b );;
gap> Display( SomeReductionBySplitEpiSummand( alpha ) );
(an empty 0 x 1 matrix)

A morphism in Category of matrices over Q
gap> Display( SomeReductionBySplitEpiSummand_MorphismFromInputRange( alpha ) );
[ [ 0 ],
  [ 1 ],
  [ -1/2 ],
  [ 1 ] ]

A morphism in Category of matrices over Q
gap> Display( SomeReductionBySplitEpiSummand_MorphismToInputRange( alpha ) );
[ [ 0, 1, 0, 0 ] ]

A morphism in Category of matrices over Q

```

2.3 Kernel

Example

```

gap> Q := HomalgFieldOfRationals();
Q
gap> V := VectorSpaceObject( 2, Q );
<A vector space object over Q of dimension 2>
gap> W := VectorSpaceObject( 3, Q );
<A vector space object over Q of dimension 3>
gap> alpha := VectorSpaceMorphism( V, [ [ 1, 1, 1 ], [ -1, -1, -1 ] ], W );
<A morphism in Category of matrices over Q>
gap> k := KernelObject( alpha );
<A vector space object over Q of dimension 1>
gap> T := VectorSpaceObject( 2, Q );
<A vector space object over Q of dimension 2>

```

```

gap> tau := VectorSpaceMorphism( T, [ [ 2, 2 ], [ 2, 2 ] ], V );
<A morphism in Category of matrices over Q>
gap> k_lift := KernelLift( alpha, tau );
<A morphism in Category of matrices over Q>
gap> HasKernelEmbedding( alpha );
false
gap> KernelEmbedding( alpha );
<A split monomorphism in Category of matrices over Q>

```

Example

```

gap> Q := HomalgFieldOfRationals();
Q
gap> V := VectorSpaceObject( 2, Q );
<A vector space object over Q of dimension 2>
gap> W := VectorSpaceObject( 3, Q );
<A vector space object over Q of dimension 3>
gap> alpha := VectorSpaceMorphism( V, [ [ 1, 1, 1 ], [ -1, -1, -1 ] ], W );
<A morphism in Category of matrices over Q>
gap> k := KernelObject( alpha );
<A vector space object over Q of dimension 1>
gap> T := VectorSpaceObject( 2, Q );
<A vector space object over Q of dimension 2>
gap> tau := VectorSpaceMorphism( T, [ [ 2, 2 ], [ 2, 2 ] ], V );
<A morphism in Category of matrices over Q>
gap> k_lift := KernelLift( alpha, tau );
<A morphism in Category of matrices over Q>
gap> HasKernelEmbedding( alpha );
false

```

Example

```

gap> Q := HomalgFieldOfRationals();
Q
gap> V := VectorSpaceObject( 2, Q );
<A vector space object over Q of dimension 2>
gap> W := VectorSpaceObject( 3, Q );
<A vector space object over Q of dimension 3>
gap> alpha := VectorSpaceMorphism( V, [ [ 1, 1, 1 ], [ -1, -1, -1 ] ], W );
<A morphism in Category of matrices over Q>
gap> k := KernelObject( alpha );
<A vector space object over Q of dimension 1>
gap> k_emb := KernelEmbedding( alpha );
<A split monomorphism in Category of matrices over Q>
gap> IsEqualForObjects( Source( k_emb ), k );
true
gap> V := VectorSpaceObject( 2, Q );
<A vector space object over Q of dimension 2>
gap> W := VectorSpaceObject( 3, Q );
<A vector space object over Q of dimension 3>
gap> beta := VectorSpaceMorphism( V, [ [ 1, 1, 1 ], [ -1, -1, -1 ] ], W );
<A morphism in Category of matrices over Q>
gap> k_emb := KernelEmbedding( beta );
<A split monomorphism in Category of matrices over Q>
gap> IsIdenticalObj( Source( k_emb ), KernelObject( beta ) );
true

```

2.4 FiberProduct

Example

```
gap> Q := HomalgFieldOfRationals();
Q
gap> A := VectorSpaceObject( 1, Q );
<A vector space object over Q of dimension 1>
gap> B := VectorSpaceObject( 2, Q );
<A vector space object over Q of dimension 2>
gap> C := VectorSpaceObject( 3, Q );
<A vector space object over Q of dimension 3>
gap> AtoC := VectorSpaceMorphism( A, [ [ 1, 2, 0 ] ], C );
<A morphism in Category of matrices over Q>
gap> BtoC := VectorSpaceMorphism( B, [ [ 1, 0, 0 ], [ 0, 1, 0 ] ], C );
<A morphism in Category of matrices over Q>
gap> P := FiberProduct( AtoC, BtoC );
<A vector space object over Q of dimension 1>
gap> p1 := ProjectionInFactorOfFiberProduct( [ AtoC, BtoC ], 1 );
<A morphism in Category of matrices over Q>
gap> p2 := ProjectionInFactorOfFiberProduct( [ AtoC, BtoC ], 2 );
<A morphism in Category of matrices over Q>
```

Index

Dimension

for IsVectorSpaceObject, [4](#)

IsVectorSpaceMorphism

for IsCapCategoryMorphism and Is-
CellOfSkeletalCategory, [4](#)

IsVectorSpaceObject

for IsCapCategoryObject and IsCellOfSkele-
talCategory, [5](#)

MatrixCategory

for IsFieldForHomalg, [3](#)

MatrixCategoryAsCategoryOfRows

for IsFieldForHomalg, [3](#)

MatrixCategoryObject

for IsMatrixCategory, IsInt, [4](#)

UnderlyingFieldForHomalg

for IsVectorSpaceMorphism, [4](#)

for IsVectorSpaceObject, [4](#)

UnderlyingMatrix

for IsVectorSpaceMorphism, [4](#)

VectorSpaceMorphism

for IsVectorSpaceObject, IsHomalgMatrix,
IsVectorSpaceObject, [3](#)

VectorSpaceObject

for IsInt, IsFieldForHomalg, [3](#)