In[1]:= `(* Author: Jurij Volčič *)`

In[2]:= `(* Last changes: 22 March 2017 *)`

# Representations of recognizable series

## Introduction

### Input and output form

We consider the rational expressions over $z[1]$, $z[2]$, etc. From the syntax point of view, we write them as in *Mathematica*'s package *NCAlgebra*, e.g.
inv[z[1]**z[2]-z[2]**z[1]]
represents the inverse of a commutator. A representation (b, A, c) corresponding to the series $b(I-\Sigma_i$ A_i $(x))^{-1}\,c$ over $M_m$(F) is (g+2)-tuple, where g is the number of variables, where b and c are 1×n and n×1 matrices of m×m blocks, respectively, and $A_i$ are n×n matrices of $m^2 \times m^2$ blocks - each representing an element of
$M_{m^2}(\mathrm{F}) \equiv M_m(\mathrm{F}) \otimes M_m(\mathrm{F})$.
Similarly, a symmetric representation (J, A, c) corresponding to the series $c^*(J-\Sigma_i$ A_i $(x))^{-1}\,c$ over $M_m$(F) is (g+2)-tuple as above, where J is a a diagonal n×n matrix of $m \times m$ blocks whose nonzero entries are ±1.

The functions work best with rational arguments (mainly due to the use of build-in function Nullspace), are pretty slow with symbolic arguments, e.g. Pi or $\sqrt{2}$, and are totally useless with numeric numbers. Therefore the user is advised to use just the first ones and the third ones combined with build-in function Rationalize. Normally, the output stays in the similar format as the input, except in the case of constructing symmetric representation, when the output is always numeric.

Instead of 1×1 matrices, we can just use their entries in the input. However, they will be presented as 1×1 matrices in the output. Also, a representation over $M_m(F)$ can be evaluated at a tuple over $M_{\mathrm{km}}(F)$.

Since our description of representations is a bit unreadable, we use special functions just to give a nice depiction of representations.

### List of main functions

★ rDim[r] ........................................................ dimension of a representation
★ RepresentationForm[r] ............................... yields a nice portrayal of an abstract representation r
★ rBuild[rf] ..................................................... excepts a rational expression rf and returns a realization of it about abstract point $(p_1,...,p_g)$, which is assumed to lie in the domain of rf

★ RepresentationForm1[r] ............................. nice form for representations with matrix coefficients
★ rBuild1[rf, p] ............................................. accepts a rational expression rf and a point p in its

domain, and returns a realization of rf about p
★ Evaluation[r, t] ........................................... given a representation r and a tuple t, returns the evaluation of r at t. If r is a realization of rf about p, then Evaluation[r,t-p] equals the evaluation of rf at p.
★ ReduceRepresentation1[r] .......................... accepts a representation r and returns a reduced representation and the dimensions of $U^L$ and $U^r$.
★ ZeroSeriesQ[r] ........................................... tests whether r represents the zero series
★ rBuild2[rf, p] ............................................... accepts a rational expression rf and a point p in its domain, and returns a minimal realization of rf about p (minimization is happening on every step, so this is the most efficient way to get minimal realizations)

★ ReprH[r] ................................................. yields the adjugate representation
★ SymRepr[r] ............................................... if r is a totally reduced representation of a Hermitian series, then this function returns its symmetric representation (the outcome is upredictable if the assumptions are not met)
★ EvaluationS[r, t] ........................................ evaluation of a symmetric representation

## NC package

In[3]:=
```
(* Actually, just the first section relies on this package. Everything
 else could be made independent of it with minor corrections,
so it might be possible to rewrite most of it in Matlab. *)

<< NC`;
<< NCAlgebra`;
```

```
You are using the version of NCAlgebra which is found in:

  ~/mathsw/NC

You can now use "<< NCAlgebra`" to load NCAlgebra or "<< NCGB`" to load NCGB.

------------------------------------------------------------
NCAlgebra - Version 5.0
Compatible with Mathematica Version 10

Authors:
  J. William Helton*
  Mauricio de Oliveira*
  Mark Stankus*
  Robert L. Miller#

* Math Dept, UCSD
# General Atomics Corp
  La  Jolla, CA 92093

Copyright:
  Helton and Miller June 1991
  Helton 2002
  Helton and de Oliveira 2016
  All rights reserved.

The program was written by the authors and by:
  David Hurst, Daniel Lamm, Orlando Merino, Robert Obar,
  Henry Pfister, Mike Walker, John Wavrik, Lois Yu,
  J. Camino, J. Griffin, J. Ovall, T. Shaheen, John Shopple.
  The beginnings of the program come from eran@slac.
  Considerable recent help came from Igor Klep.

This program was written with support from
  AFOSR, NSF, ONR, Lab for Math and Statistics at UCSD,
  UCSD Faculty Mentor Program,
  and US Department of Education.
  Primary support in 2010 is from the
    NSF Division of Mathematical Sciences.

If you
  (1) are a user,
  (2) want to be a user,
  (3) refer to NCAlgebra in a publication, or
  (4) have had an interesting experience with NCAlgebra,
let us know by sending an e-mail message to

  ncalg@math.ucsd.edu.

We do not want to restrict access to NCAlgebra, but do
  want to keep track of how it is being used.

For NCAlgebra updates see:

  www.math.ucsd.edu/~ncalg
  www.github.com/NCAlgebra/NC

------------------------------------------------------------
```

# Symbolic representations

# Concrete realizations

# Minimization

# Zero series test

# Symmetric representations

# Examples

In[71]:=
```
SNC[z, p];
```

In[72]:=
```
rf = inv[z[1]] ** z[2] ** inv[z[1] ** z[2] - z[2] ** z[1]];
point = { {{0, - 1}, {1, 0}}, {{0, 0}, {1, 0}}};
r = rBuild1[rf, point];
rr = ReduceRepresentation1[r] // First;
rr // RepresentationForm1
```

Out[76]//TableForm=

$$\begin{pmatrix} \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 \\ -1 & 0 \end{pmatrix} & 0 & \begin{pmatrix} -1 & 0 \\ 0 & 0 \end{pmatrix} \end{pmatrix} \quad \begin{pmatrix} x.\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} & x.\begin{pmatrix} 0 & 0 \\ -1 & 0 \end{pmatrix} & 0 & x.\begin{pmatrix} -1 & 0 \\ 0 & 0 \end{pmatrix} \\ 0 & 0 & 0 & 0 \\ 0 & x.\begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix} & x.\begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} & x.\begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \\ 0 & x.\begin{pmatrix} -1 & 0 \\ 0 & 0 \end{pmatrix} & 0 & x.\begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \end{pmatrix}$$

In[77]:=

```
rf = inv[1 - inv[z[1]] ** z[2] ** inv[z[1] ** z[2] - z[2] ** z[1]] -
    inv[z[1] ** z[2] - z[2] ** z[1]] ** z[1] ** inv[z[1] ** z[2] - z[2] ** z[1]]];
point = { {{0, - 1}, {1, 0}}, {{0, 0}, {1, 0}}};
r = rBuild2[rf, point];
r // RepresentationForm1
r[[2]][[2]][[1]] // MatrixForm
```

Out[80]//TableForm=

$$\left(\begin{pmatrix} -\frac{1}{3} & -\frac{1}{3} \\ -\frac{2}{3} & \frac{1}{3} \end{pmatrix} \begin{pmatrix} \frac{1}{3} & 0 \\ \frac{2}{3} & 0 \end{pmatrix} \begin{pmatrix} -\frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{2}{3} \end{pmatrix} \begin{pmatrix} -\frac{1}{3} & \frac{1}{3} \\ -\frac{2}{3} & -\frac{1}{3} \end{pmatrix} \begin{pmatrix} -\frac{2}{3} & -\frac{1}{3} \\ -\frac{1}{3} & -\frac{2}{3} \end{pmatrix} \begin{pmatrix} 0 & \frac{1}{3} \\ 0 & \frac{2}{3} \end{pmatrix} \begin{pmatrix} -\frac{2}{3} & \frac{1}{3} \\ -\frac{1}{3} & -\frac{1}{3} \end{pmatrix} \begin{pmatrix} \frac{1}{3} & \frac{1}{3} \\ -\frac{1}{3} & \frac{2}{3} \end{pmatrix}\right)$$

Out[81]//MatrixForm=

$$\begin{pmatrix} -\frac{2}{3} & -\frac{2}{3} & 0 & 0 \\ -\frac{2}{3} & \frac{1}{3} & 0 & 0 \\ 0 & 0 & -\frac{2}{3} & -\frac{2}{3} \\ 0 & 0 & -\frac{2}{3} & \frac{1}{3} \end{pmatrix}$$

In[82]:=

```
rf = inv[z[1] ** z[2] - z[2] ** z[1]];
point = { {{0, 1}, {0, 0}}, {{0, 0}, {1, 0}}};
r = rBuild1[rf, point];
rr = ReduceRepresentation1[r] // First;

testfun2[tuple_] := {Evaluation[r, tuple - point] -
     Inverse[tuple[[1]].tuple[[2]] - tuple[[2]].tuple[[1]]],
    Evaluation[rr, tuple - point] - Inverse[
      tuple[[1]].tuple[[2]] - tuple[[2]].tuple[[1]]]};

Do[
  first = RandomInteger[{-10, 10}, {2, 2}] / RandomChoice[Range[-11, 11, 2]];
  second = RandomInteger[{-10, 10}, {2, 2}] / RandomChoice[Range[-11, 11, 2]];
  Print[Norm /@ testfun2[{first, second}]],
  {10}
 ];
```

```
{0, 0}
{0, 0}
{0, 0}
{0, 0}
{0, 0}
{0, 0}
{0, 0}
{0, 0}
{0, 0}
{0, 0}
```

In[88]:=

```
posdimzero = {
   {{{{1, 0}, {0, 0}}}},
   {{KroneckerProduct[{{0, 0}, {0, 1}}, id[2]]}},
   {{{{0, 0}, {0, 1}}}}
  };
ZeroSeriesQ[posdimzero] // Print;
ZeroSeriesQ[rBuild1[z[1], {0}]] // Print;
ZeroSeriesQ[rPlus1[rBuild1[z[1], {0}], rBuild1[-z[1], {0}]]] // Print;
```

```
True
False
True
```

In[92]:=

```
rf = z[1] ** inv[z[2] - 1 / 2] + inv[z[2] - 1 / 2] ** z[1];
point = {0, 0};
r = rBuild1[rf, point];
rr = ReduceRepresentation1[r] // First;
rsym = SymRepr[rr];
rsym // RepresentationForm1 // Print;
symmat = {{0, 1}, {1, 0}};
Norm[
   EvaluationS[rsym, {symmat, symmat}] -
    (symmat.Inverse[symmat - 1 / 2 * id[2]] + Inverse[symmat - 1 / 2 * id[2]].symmat)
  ] // Print;
```

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \quad \begin{pmatrix} -0.724564\,x & -0.724564\,x & 0.613941\,x & -0.613941\,x \\ -0.724564\,x & -0.724564\,x & 0.613941\,x & -0.613941\,x \\ 0.613941\,x & 0.613941\,x & 0.169863\,x & -0.169863\,x \\ -0.613941\,x & -0.613941\,x & -0.169863\,x & 0.169863\,x \end{pmatrix} \quad \begin{pmatrix} 1.86824\,x \\ 2.22045 \times 10^{-15}\,x \\ 0.632456\,x \\ 0.392232\,x \end{pmatrix}$$

$3.34905 \times 10^{-15}$

In[100]:=

```
rf = inv[z[1] + 1] ** z[2] ** inv[z[1] ** z[2] - z[2] ** z[1]] -
   inv[z[1] ** z[2] - z[2] ** z[1]] ** z[1] **
    inv[inv[z[1]] ** z[2] - z[2] ** inv[z[1]]];
point = { {{0, - 1}, {1 / 2, 0}}, {{0, 0}, {1, 0}}};
r = rBuild2[rf, point];
r // RepresentationForm1
```

Out[103]//TableForm=

$$\left( \begin{pmatrix} 0 & 1 \\ \frac{1}{2} & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 0 \\ -1 & 0 \end{pmatrix} \quad \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad \frac{1}{2} \quad \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & \frac{1}{2} \\ -\frac{1}{4} & 0 \end{pmatrix} \quad \begin{pmatrix} \frac{2}{3} & \frac{2}{3} \\ -\frac{1}{3} & \frac{2}{3} \end{pmatrix} \quad \begin{pmatrix} \frac{2}{3} & 0 \\ -\frac{1}{3} & 0 \end{pmatrix} \quad 0 \quad \begin{pmatrix} -\frac{2}{3} & 0 \\ -\frac{2}{3} & 0 \end{pmatrix} \right)$$

In[104]:=

```
rf =
  inv[z[2] ** inv[z[1] ** z[2] - z[2] ** z[1]] - inv[z[1] ** z[2] - z[2] ** z[1]] **
       z[1]] ** inv[z[2] ** inv[z[1] ** z[2] - z[2] ** z[1]] -
      inv[z[1] ** z[2] + z[2] ** z[1]] ** z[1]] -
   inv[z[2] ** inv[z[1] ** z[2] - z[2] ** z[1]] -
      inv[z[1] ** z[2] + z[2] ** z[1]] ** z[1]] **
    inv[z[2] ** inv[z[1] ** z[2] - z[2] ** z[1]] -
      inv[z[1] ** z[2] - z[2] ** z[1]] ** z[1]];
point = { {{0, - 1}, {0, 0}}, {{0, 0}, {1, 0}}};
r = rBuild2[rf, point];
r // RepresentationForm1
```

Out[107]//TableForm=

$$\left( \begin{pmatrix} -1 & 0 \\ 0 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \quad 0 \quad \begin{pmatrix} -1 & 0 \\ 0 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \quad -1 \quad \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \quad \begin{pmatrix} -1 & 0 \\ 0 & 0 \end{pmatrix} \quad 0 \quad \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \right)$$