A SOFTWARE PACKAGE TO COMPUTE AUTOMORPHISMS OF GRADED ALGEBRAS

SIMON KEICHER

Abstract. We present a library autgradalg.lib for the free computer algebra system Singular to compute automorphisms of integral, finitely generated C-algebras that are graded pointedly by a finitely generated abelian group. It implements the algorithms developed in [10]. We apply the algorithms to Mori dream spaces and investigate the automorphism groups of a series of Fano varieties.

1. Introduction and setting

Consider an integral, finitely generated C-algebra $R$ that is graded by a finitely generated abelian group $K$, i.e., we have a decomposition

$$R = \bigoplus_{w \in K} R_w$$

with $ff' \in R_{w+w'}$ for all $f \in R_w, f' \in R_{w'}$.

Let the grading to be effective, i.e., the monoid $\vartheta R \subseteq K$ of all $w \in K$ with $R_w \neq \{0\}$ generates $K$ as a group, and pointed: this means that we have $R_0 = \mathbb{C}$ and the polyhedral cone in $K \otimes \mathbb{Q}$ generated by $\vartheta R$ is pointed.

We are interested in the automorphism group $Aut_K(R)$: it consists of all pairs $(\varphi, \psi)$ such that $\varphi: R \rightarrow R$ is an automorphism of C-algebras, $\psi: K \rightarrow K$ is an automorphism of groups and $\varphi(R_w) = R_{\psi(w)}$ holds for all $w \in K$. Note that $Aut_K(R)$ not only is an important invariant of the algebra $R$, the methods to compute it can by applied to compute symmetries of homogeneous ideals $I$. Once given explicitly, the knowledge of the latter largely accelerates further computations involving $I$, see [11, 5, 14] for examples.

This note presents an implementation autgradalg.lib of the algorithms from [10] to compute $Aut_K(R)$. It is written for the free computer algebra system Singular [7] and is available at [13]. In Section 2, we describe the algorithm [10] to compute $Aut_K(R)$ and explain our implementation by a series of examples. In Section 3, we apply our implementation to Mori dream spaces. As a result, we determine in Proposition 3.1 information on the automorphism groups of a class of Fano threefolds listed in [9].

2. Automorphisms of graded algebras

Let us fix the assumptions on the algebra $R$ for our algorithms. Firstly, we assume the grading group $K$ to be of shape $\mathbb{Z}^k \oplus \mathbb{Z}/a_1\mathbb{Z} \oplus \ldots \oplus \mathbb{Z}/a_l\mathbb{Z}$. In particular, $k$ and the list $a_1, \ldots, a_l \in \mathbb{Z}_{>1}$ encode $K$. The $K$-grading is determined by the degree matrix $Q = [q_1, \ldots, q_r]$ which has the $q_i := \deg(T_i)$ as its columns. Moreover, we expect $R$ to be given explicitly in term of generators and relations:

$$R = S/I, \quad S := \mathbb{C}[T_1, \ldots, T_r] \quad I := \langle g_1, \ldots, g_s \rangle \subseteq S.$$

As one can remove linear equations, it is no restriction to assume that $R$ is minimally presented, i.e., $I \subseteq (T_1, \ldots, T_r)^2$ holds and the generating set $\{g_1, \ldots, g_s\}$ for $I$ is minimal. From an implementation point of view, it is convenient to impose the following slight restrictions:

2010 Mathematics Subject Classification. 13P10, 14Q15, 14J50, 13A02, 14L30, 13A50.
the homogenous components $I_{q_1},\ldots,I_{q_r}$ are all trivial,

- the set $\{q_{i,0},\ldots,q_{i,k}\} \subseteq \mathbb{Z}^k$ of the free parts $q_{i}^0 \in \mathbb{Z}^k$ of the $q_i$ contains a lattice basis for $\mathbb{Z}^k$.

**Example 2.1 (autgradalg.lib).** Consider the following $K := \mathbb{Z}^3 \oplus \mathbb{Z}/2\mathbb{Z}$-graded $\mathbb{C}$-algebra $R$ from \cite{10, Example 2.1] where

$$R = S/I, \quad S := \mathbb{C}[T_1,\ldots,T_5], \quad I := \langle T_1 T_6 + T_2 T_3 + T_3 T_4 + T_7 T_8 \rangle,$$

$$Q := \begin{bmatrix}
1 & 1 & 0 & 0 & -1 & -1 & 2 & -2 \\
0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 & 1 & 1 & 1 & 1
\end{bmatrix}.$$ 

Then the $K$-grading given by $Q$ is effective and pointed as hinted in the following picture. To use autgradalg.lib, download it from \cite{13} and start Singular in the same directory. We enter $R$ with the commands

```
> LIB "gfanlib.so"; // for cones
> LIB "new_autgradalg.lib";
> intmat Q[4][8] =
1 1 0 0 -1 -1 2 -2
0 1 1 1 1 1 1 1
0 1 1 1 1 1 1 1
1 0 1 1 1 1 1 1;
> list TUR = 2; // torsion part of K
> ring S = 0,T(1..8),dp;
> setBaseMultigrading(Q); // grading
```

Let us recall shortly the steps of the algorithm to compute $\text{Aut}_K(R)$; for details, we refer to \cite{10}. The overall idea is to present $\text{Aut}_K(R)$ as a stabilizer in the automorphism group $\text{Aut}_K(S)$ of the $K$-graded polynomial ring $S$. In a first step, we will compute a presentation $\text{Aut}_K(S) \subseteq \text{GL}(n)$ for some $n \in \mathbb{Z}_{\geq 1}$. The set $\Omega_S := \{q_1,\ldots,q_r\}$ of generator weights will play a major role. We make use of the following $\text{GL}(n)$-action.

**Construction 2.2.** See \cite{10, Construction 3.3]. Write $\Omega_S = \{w_1,\ldots,w_s\}$. Determine a $\mathbb{C}$-vector space basis $B_i$ for $S_w$, consisting of monomials. Then the concatenation $B := (B_1,\ldots,B_s)$ is a basis for $V = \bigoplus_i S_w$. With $n := |B|$, in terms of $B$, each $A \in \text{GL}(n)$ defines a linear map $\varphi_A : V \to V$. We obtain an algebraic action

$$\text{GL}(n) \times S \to S, \quad (A,f) \mapsto A \cdot f := f(\varphi_A(T_1),\ldots,\varphi_A(T_r)).$$

For the second step, the idea is to determine equations cutting out those matrices in $\text{GL}(n)$ that permute the homogeneous components $S_w$ of same dimension where $w \in \Omega_S$. As $\Omega_S$ must be fixed by each automorphism, it suffices to consider the finite set

$$\text{Aut}(\Omega_S) := \{\psi \in \text{Aut}(K); \psi(\Omega_S) = \Omega_S\} \subseteq \text{Aut}(K).$$

It can be computed by tracking a lattice basis among the set of free parts $q_{i,0}^0$ of the $q_i$, see \cite{10, Remark 3.1].

**Algorithm 2.3 (Compute $\text{Aut}_K(S)$).** See \cite{10, Algorithm 3.7]. Input: the $K$-graded polynomial ring $S$.

- Determine $\Omega_S = \{w_1,\ldots,w_s\}$. Compute a basis $B$ as in Construction \cite{10}.
- Define the polynomial ring $S' := \mathbb{C}[Y_{ij}; 1 \leq i,j \leq n]$.
- Compute an ideal $J \subseteq S'$ whose equations ensure for each $A \in V(J) \subseteq \text{GL}(n)$ the multiplicative condition $A \cdot (f_1 f_2) = (A \cdot f_1)(A \cdot f_2)$ where $f_i \in S$.
- Compute $\text{Aut}(\Omega_S) \subseteq \text{Aut}(K)$. Determine the subset $\Gamma_0 \subseteq \text{Aut}(\Omega_S)$ of those $B$, that map $B_i$ bijectively to $B_j$ where $w_j = B \cdot w_i$.
- For each $B \in \Gamma_0$, do
  - compute an ideal $J_B \subseteq S'$ ensuring that each matrix in $V(J_B) \subseteq \text{GL}(n)$ maps the component $S_{w}$ to the component $S_{B \cdot w}$ where $w \in \Omega_S$. 

Example 2.5 (autgradalg.lib II). Let us apply Algorithm 2.3 to Example 2.1. Here, $\mathcal{B} = (T_1, \ldots, T_8)$ and all bases $\mathcal{B}_i = (T_i)$ are one-dimensional. Since no weight appears multiple times, $\Omega_S = \{q_1, \ldots, q_8\}$. Next, the algorithm will compute $\text{Aut}(\Omega_R)$. In our implementation one can also trigger this step manually if desired:

```plaintext
> list origs = autGenWeights(Q, TOR);
> setring Sprime;
> def Sprime = autKS(TOR);
> print(listAutKS[2][3][2]);
```

The result, origs, is a list of four integral matrices (intmats) standing for the automorphisms of the generator weights

\[
\begin{align*}
\text{Aut}(\Omega_S) &= \left\{ \text{id}, \begin{bmatrix} 1 & -2 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}, \\
& \quad \begin{bmatrix} -1 & 2 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \right\}.
\end{align*}
\]

Note that Aut($\Omega_R$) is isomorphic to the symmetry group $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$ of a 2-dimensional rhombus. We now compute Aut$_K(S)$ with the command

```plaintext
> def Sprime = autKS(TOR);
> setring Sprime;
```

A closer inspection shows that $\text{Sprime}$ stands for the ring $\mathcal{S}' = \mathbb{Q}[Y_1, \ldots, Y_{64}, Z]$. Furthermore, a list listAutKS will be exported: each element is a triple $(A_B, B, J_B)$ where $B$ runs through the four elements of Aut($\Omega_R$) and $A_B$ is a formal matrix over $\text{Sprime}$ that encodes isomorphisms of $S$ as in Remark 2.4(iii). For instance, for listAutKS[2], the second entry in the triple $(A_B, B, J_B)$ is the second matrix listed in (1) and the matrix $A_B$ is

```plaintext
> print(listAutKS[2][1][1]);
```

\[
\begin{bmatrix}
Y(1) & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

The equations obtained from the zero-entries in $A_B$ and its invertible-condition are stored in the ideal $J_B$. The third entry is:

```plaintext
> print(listAutKS[2][3]);
```

\[
\begin{align*}
Y(2), Y(3), \ldots, Y(63), \ Y(64), \ -Y(1)Y(13)Y(24)Y(31)Y(34)Y(46)Y(52)Y(59)Z - 1.
\end{align*}
\]

Moreover, an ideal Iexported, called $J$ in in Algorithm 2.3, is being exported that is the product over all the ideals $J_B$ where $B$ runs through Aut($\Omega_R$). This means Aut$_K(S) \cong S'/J$ is isomorphic to $\text{Sprime}$ modulo Iexported; the degree matrix of $\text{Sprime}$ can be obtained via getVariableWeights().

We come to Aut$_K(R)$. Restricting the group action of Construction 2.2 to Aut$_K(S) \subseteq \text{GL}(n)$, we have an algebraic subgroup given as stabilizer

\[
\text{Stab}_I(\text{Aut}_K(S)) := \{ A \in \text{Aut}_K(S); A \cdot I = I \} \subseteq \text{Aut}_K(S).
\]
Provided \( I_w = \{0\} \) holds for all \( w \in \Omega_S \), in \cite{10} the authors have shown that we have an isomorphism

\[
\text{Stab}_I(\text{Aut}_K(S)) \cong \text{Aut}_K(R).
\]

The final step then is the following. Define the set \( \Omega_I := \{\deg(g_1), \ldots, \deg(g_n)\} \) of ideal generator degrees. The idea is to compute (linear) equations ensuring that the vector spaces \( I_u \), where \( u \in \Omega_I \), are mapped to one-another.

**Algorithm 2.6 (Computing \( \text{Aut}_K(R) \)).** See \cite{10} Algorithm 3.8. *Input*: the \( K \)-graded polynomial ring \( S \) and the defining ideal \( I \subseteq S \) of \( R \).

- Let \( J \subseteq S' := \mathbb{C}[Y_{ij}] \) be the output of Algorithm 2.3.
- Compute \( \Omega_I \) and form the \( \mathbb{C} \)-vector space \( W := \bigoplus_{I_u} S_u \).
- For the vector space \( I_W = I \cap W \subseteq W \), compute
  - a \( \mathbb{C} \)-basis \( \{h_1, \ldots, h_l\} \) and
  - a description \( I_W = V(\ell_1, \ldots, \ell_m) \) with linear forms \( \ell_i \in W^* \).
- With the \( \text{GL}(n) \)-action from Construction 2.2 and \( Y = (Y_{ij}) \), we obtain the ideal \( J' := \langle \ell_i(Y \cdot h_j) \rangle ; 1 \leq i \leq m, 1 \leq j \leq l \rangle \subseteq S' \).

*Output*: the ideal \( J + J' \subseteq S' \). Then \( V(J + J') \subseteq \text{GL}(n) \) is an algebraic subgroup isomorphic to \( \text{Aut}_K(R) \).

**Remark 2.7.**

(i) Algorithms 2.4 and 2.3 do not make use of Gröbner basis computations. However, in *Singular*, it usually is quicker to compute \( J \cap J_B \) instead of \( J \cdot J_B \).

(ii) Computing \( G := \text{Aut}_K(R) \subseteq \text{GL}(n) \) with Algorithm 2.6 enables us to directly compute the number of irreducible components \( |G : G^0| \) and the dimension of \( G \) by Gröbner basis computations.

**Example 2.8 (autgradalg.lib III).** Continuing Example 2.5, let us compute \( \text{Aut}_K(R) \). We first switch back to \( S \), enter the defining ideal \( I \) for \( R = S/I \) and start the computation of \( \text{Aut}_K(R) \):

```plaintext
> setring S;
> ideal I = T(1)*T(6) + T(2)*T(5) + T(3)*T(4) + T(7)*T(8);
> def Sres = autGradAlg(I, TOR);
> setring Sres;
```

The resulting ring \( S_{\text{res}} \) is identical to \( S_{\text{prime}} \). A list \text{stabExported} is being exported; the interpretation of the entries is identical to that of the list \text{listAutKS} from Example 2.3 with the difference, that the ideal part now contains additional equations describing the stabilizer: for example

```plaintext
> stabExported[2][3]
Y(2), Y(3), \ldots, Y(63), Y(64), \ldots, Y(1)Y(13)Y(24)Y(31)Y(34)Y(46)Y(52)Y(59)Z - 1,
\ldots
Y(24)Y(31) + Y(52)Y(59), X(13)Y(34) - Y(52)Y(59), \ldots
```

Moreover, an ideal \text{Jexported} is being exported that is the product over all \( J_B \) as before. Then \( S_{\text{res}} \) modulo \text{Jexported} is isomorphic to \( \text{Aut}_K(R) \). The grading is obtained as before with \text{getVariableWeights}().

3. **Application: Mori Dream Spaces**

In this section, we shortly recall from \cite{10} how the algorithms from the last section can be applied to a class of varieties in algebraic geometry.

To a normal algebraic variety \( X \) over \( \mathbb{C} \) with finitely generated class group \( \text{Cl}(X) \) one can assign a \( \text{Cl}(X) \)-graded \( \mathbb{C} \)-algebra, its so-called *Cox ring*

\[
\text{Cox}(X) = \bigoplus_{D \in \text{Cl}(X)} \Gamma(X, \mathcal{O}(D)),
\]

see e.g. \cite{1} for details on this theory. If \( X \) is finitely generated, \( X \) is called a *Mori dream space*. For example, each toric variety or each smooth Fano variety is a Mori
dream space \([6, 4]\). The Cox ring has strong implications on the underlying Mori dream space. More precisely, \(X\) can be recovered as a good quotient

\[
\operatorname{Spec}(R) =: X \supseteq \tilde{X} \xrightarrow{\psi_H} X
\]
of an open subset \(\tilde{X}\) by the characteristic quasitorus \(H := \operatorname{Spec}(\mathbb{C}[K])\). In fact, \(\tilde{X}\) is determined by an ample class \(w \in \text{Cl}(X)\). This opens up a computer algebra based approach \([9, 12]\) to Mori dream spaces. In \([2]\), it has been shown that \((2)\) translates to automorphisms of \(X\) as follows:

\[
\text{Aut}_{\text{Cl}(X)}(\operatorname{Cox}(X)) \cong \text{Aut}_H(\tilde{X}) \supseteq \text{Aut}_H(\tilde{X}) \xrightarrow{H} \text{Aut}(X)
\]

Here, by \(\text{Aut}_H(Y)\) we mean the group of \(H\)-equivariant automorphisms of \(Y\); these are pairs \((\varphi, \psi)\) with \(\varphi: Y \rightarrow Y\) being an automorphism of varieties and \(\psi: H \rightarrow H\) an automorphism of affine algebraic groups such that \(\varphi(h \cdot y) = \psi(h) \cdot y\) holds for all \(h \in H\) and \(y \in Y\). By \([3]\), we directly can compute \(\text{Aut}_H(\tilde{X})\) with Algorithm \([2, 3]\). In the next proposition, we investigate the symmetries of the list of Fano varieties \([3]\).

**Proposition 3.1.** Let \(X_i\) be the non-toric terminal Fano threefold of Picard number one with an effective two-torus action from the classification \([3, \text{Theorem 1.1}]\).

(i) For all \(1 \leq i \leq 41\), Algorithm \([2, 3]\) is able to compute a presentation of \(G_i := \text{Aut}_H(\tilde{X}_i)\) as an affine algebraic subgroup \(V(J_i) \subseteq \text{GL}(n_i)\).

(ii) Using (i), we list the dimensions \(\dim(G_i)\) and the number of components \([G_i : G_0]\) of the following \(G_i \subseteq \text{GL}(n_i)\):

\[
\begin{array}{cccc}
X_3 & \text{Aut}(\Omega S) & \text{dim}(G_i) & [G_i : G_0] \\
X_6 & \{1\} & 5 & 4 \\
X_7 & \{1\} & 5 & 4 \\
X_{10} & \{1\} & 4 & 1 & 3 \\
X_{12} & \{1\} & 6 & 3 \\
X_{13} & \{1\} & 4 & 1 & 3 \\
X_{14} & \{1\} & 3 & 2 \\
X_{15} & \{1\} & 5 & 4 \\
X_{16} & \{1\} & 3 & 2 \\
X_{18} & \{1\} & 6 & 5 \\
X_{19} & \{1\} & 4 & 1 & 3 \\
X_{20} & \{1\} & 5 & 4 \\
X_{21} & \{1\} & 3 & 2 & 2 \\
X_{25} & \{1\} & 4 & 1 & 3 \\
X_{26} & \{1\} & 3 & 2 \\
X_{28} & \{1\} & 4 & 1 & 3 \\
X_{33} & \{1\} & 6 & 2 & 5 \\
X_{34} & \{1\} & 6 & 2 & 5 \\
X_{36} & \{1\} & 5 & 1 & 4 \\
X_{37} & \{1\} & 4 & 2 & 3 \\
X_{38} & \{1\} & 4 & 3 & 3 \\
X_{39} & \{1\} & 3 & 2 \\
X_{40} & \{1\} & 3 & 1 & 2 \\
X_{42} & \{1\} & 3 & 2 \\
X_{45} & \{1\} & 4 & 1 & 3 \\
X_{46} & \{1\} & 4 & 1 & 3 \\
X_{47} & \{1\} & 3 & 1 & 2 \\
\end{array}
\]

**Proof.** This is an application of Algorithm \([2, 3]\) and of the \textsc{Singular} commands to compute dimension and absolute components, see for example \([8]\). We performed the computations on an older machine (Intel celeron CPU, 4 GB Ram) and cancelled them after several seconds. The files are available at \([13]\). \(\square\)
In [10], the authors have also presented algorithms to compute \( \operatorname{Aut}_H(\hat{X}) \) and generators for the Hopf algebra \( \mathcal{O}(\operatorname{Aut}(X)) \). Both algorithms are also implemented in our library. However, the case \( \mathcal{O}(\operatorname{Aut}(X)) \) involves a Hilbert basis computation that usually renders the computation infeasible. We therefore finish this note with an example.

**Example 3.2** (**autgradalg.lib** IV). In Example 2.8 the algebra \( R \) is the Cox ring of a Mori dream space: fix an ample class, say \( w := (0, 0, 2) \in K \otimes \mathbb{Q} \), then \( R \) and \( w \) define a Mori dream space \( X = X(R, w) \). The characteristic quasitorus is \( H = (\mathbb{C}^*)^3 \times \{ \pm 1 \} \).

In 2.8 we have already computed \( \operatorname{Aut}_H(X) \cong G := \operatorname{Aut}_K(R) \). From it, we obtain \( \operatorname{Aut}_H(\hat{X}) \) as follows: first, \( w \) defines a certain polyhedral cone, the GIT-cone \( \lambda(w) \). Then \( \operatorname{Aut}_H(\hat{X}) \) is obtained from \( G \) by choosing only those elements \((A_B, B, J_B)\) of the list \( \text{stabExported} \) where \( B \in \operatorname{Aut}(\Omega_S) \) fixes \( \lambda(w) \). In our library, you can compute it with (making use of \text{gitfan.lib} [5])

\[
\begin{align*}
> \text{intvec } w & = 1,9,16,0; \text{ // drawn in blue} \\
> \text{setring } R; \text{ // from before} \\
> \text{def } RR & = \text{autXhat}(I, w, \text{TOR}); \\
> \text{setring } RR;
\end{align*}
\]

Then a list \( \text{RES} \) will be exported; it is identical to the list \( \text{stabExported} \) from Example 2.8 with the difference, that it contains only the element \( \text{stabExported}[1] \) as the other matrices \( B \) do not fix \( \lambda(w) \). The computation of generators for \( \mathcal{O}(\operatorname{Aut}(X)) \) is not feasible here; in principle, the command is \( \text{autX}(I, w, \text{TOR}) \).

**References**

[1] I. Arzhantsev, U. Derenthal, J. Hausen, and A. Laface. *Cox rings*, volume 144 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, Cambridge, 2014.

[2] I. Arzhantsev, J. Hausen, E. Herppich, and A. Liendo. The automorphism group of a variety with torus action of complexity one. *Mosc. Math. J.*, 14:429–471, 2014.

[3] B. Bechtold, J. Hausen, E. Huggenberger, and M. Nicolussi. On terminal Fano 3-folds with 2-torus action. *International Mathematics Research Notices*, 5:1563–1602, 2016.

[4] C. Birkar, P. Cascini, C. D. Hacon, and J. McKernan. Existence of minimal models for varieties of log general type. *Journal of the American Mathematical Society*, 23(2):405–468, 2010.

[5] J. B"ohm, S. Keicher, and Y. Ren. Computing GIT-fans with symmetry and the Mori chamber decomposition of \( \overline{M}_{0,6} \). 2016. Preprint. See arXiv:1603.09241.

[6] D. A. Cox. The homogeneous coordinate ring of a toric variety. *J. Algebraic Geom.*, 4(1):17–50, 1995.

[7] W. Decker, G.-M. Greuel, G. Pfister, and H. Sch"onemann. *SINGULAR 4-1-0 — A computer algebra system for polynomial computations*. http://www.singular.uni-kl.de, 2016.

[8] G.-M. Greuel and G. Pfister. *A Singular Introduction to Commutative Algebra*. Springer Publishing Company, Incorporated, 2nd edition, 2007.

[9] J. Hausen and S. Keicher. A software package for Mori dream spaces. *LMS Journal of Computation and Mathematics*, 18(1):647–659, 2015.

[10] J. Hausen, S. Keicher, and R. Wolf. Computing automorphisms of Mori dream spaces. *Mathematics of Computation*, to appear.

[11] A. N. Jensen. *Gfan*, a software system for Gr"obner fans and tropical varieties. Available at http://home.imf.au.dk/jensen/software/gfan/gfan.html.

[12] S. Keicher. *Algorithms for Mori Dream Spaces*. PhD thesis, Universität Tübingen, 2014.

[13] S. Keicher. *autmds.lib* — a library for *Singular* to compute automorphisms of graded algebras, 2017. Will be made available at http://www.math.uni-tuebingen.de/user/keicher/autgradlib/

[14] S. Steidel. *Gr"obner bases of symmetric ideals*. *J. Symbolic Comput.*, 54:72–86, 2013.

Mathematisches Institut, Universität Tübingen, Auf der Morgenstelle 10, 72076 Tübingen, Germany

E-mail address: keicher@mail.mathematik.uni-tuebingen.de