```
gap> g:= SymmetricGroup( 4 );
Sym( [ 1 .. 4 ] )
gap> tbl:= CharacterTable( g );;  HasIrr( tbl );
false
gap> tblmod2:= CharacterTable( tbl, 2 );
BrauerTable( Sym( [ 1 .. 4 ] ), 2 )
gap> tblmod2 = CharacterTable( tbl, 2 );
true
gap> tblmod2 = BrauerTable( tbl, 2 );
true
gap> libtbl:= CharacterTable( "M" );
CharacterTable( "M" )
gap> BrauerTable( libtbl, 2 );
fail
gap> CharacterTable( "Symmetric", 4 );
CharacterTable( "Sym(4)" )
gap> ComputedBrauerTables( tbl );
[ , BrauerTable( Sym( [ 1 .. 4 ] ), 2 ) ]
```

```
i5 : betti(t,Weights=>{1,0})

            0 1  2  3 4
o5 = total: 1 4 13 14 4
         0: 1 .  .  . .
         1: . 2  2  4 2
         2: . 2  5  6 .
         3: .  .  4  . 2
         4: .  .  .  4 .
         5: .  2  .  . .

o5 : BettiTally
i6 : betti(t,Weights=>{0,1})

            0 1  2  3 4
o6 = total: 1 4 13 14 4
         0: 1 .  .  . .
         1: . 2  2  4 2
         2: . 2  5  6 .
         3: .  .  4  . 2
         4: .  .  .  4 .
         5: .  2  .  . .

o6 : BettiTally
i7 : t1 = betti(t,Weights=>{1,1})

            0 1  2  3 4
o7 = total: 1 4 13 14 4
         0: 1 .  .  . .
         1: .  .  .  . .
         2: .  .  .  . .
         3: . 2  .  . .
         4: .  .  .  . .
         5: . 2  .  . .
         6: .  .  1  . .
         7: .  .  8  6 .
         8: .  .  4  8 4

o7 : BettiTally
i8 : peek t1

o8 = BettiTally{(0, {0, 0}, 0) => 1 }
                (1, {2, 2}, 4) => 2
                (1, {3, 3}, 6) => 2
                (2, {3, 7}, 10) => 2
                (2, {4, 4}, 8) => 1
                (2, {4, 5}, 9) => 4
                (2, {5, 4}, 9) => 4
                (2, {7, 3}, 10) => 2
                (3, {4, 7}, 11) => 4
                (3, {5, 5}, 10) => 1
                (3, {7, 4}, 11) => 4
                (4, {5, 7}, 12) => 2
                (4, {7, 5}, 12) => 2
```

```
ring r1 = 32003,(x,y,z),ds;
int a,b,c,t=11,5,3,0;
poly f = x^a+y^b+z^(3*c)+x^(c+2)*y^(c-1)+x^
         x^(c-2)*y^c*(y^2+t*x)^2;
option(noprot);
timer=1;
ring r2 = 32003,(x,y,z),dp;
poly f=imap(r1,f);
ideal j=jacob(f);
vdim(std(j));
==> 536
vdim(std(j+f));
==> 195
timer=0;  // reset timer
```

Journal of Software for Algebra and Geometry

A software package to compute automorphisms of graded algebras

SIMON KEICHER

vol 8 2018

JSAG **8** (2018), 11–19
dx.doi.org/10.2140/jsag.2018.8.11

The Journal of Software for
Algebra and Geometry

A software package to compute automorphisms of graded algebras

SIMON KEICHER

ABSTRACT: We present `autgradalg.lib`, a Singular library to compute automorphisms of integral, finitely generated $\mathbb{C}$-algebras that are graded pointedly by a finitely generated abelian group. The library implements algorithms of Hausen, Keicher and Wolf (*Math. Comp.* **86** (2017), 2955–2974). We apply these to Mori dream spaces and investigate the automorphism groups of a series of Fano varieties.

1. INTRODUCTION AND SETTING. Consider an integral, finitely generated $\mathbb{C}$-algebra R that is graded by a finitely generated abelian group K; i.e., we have a decomposition

$$R = \bigoplus_{w \in K} R_w \quad \text{with } ff' \in R_{w+w'} \text{ for all } f \in R_w,\ f' \in R_{w'}.$$

Let the grading be *effective* (so that 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 ϑ_R is pointed.

We are interested in the *automorphism group* $\mathrm{Aut}_K(R)$: it consists of all pairs (φ, ψ) such that $\varphi : R \to R$ is an automorphism of $\mathbb{C}$-algebras, $\psi : K \to K$ is an automorphism of groups and $\varphi(R_w) = R_{\psi(w)}$ holds for all $w \in K$. Not only is $\mathrm{Aut}_K(R)$ an important invariant of the algebra R, but the methods used to compute it can be applied to compute symmetries of homogeneous ideals I. Once given explicitly, knowledge of these symmetries accelerates further computations involving I; see [Jensen 2017; Boehm et al. 2016; Steidel 2013] for examples.

This article introduces `autgradalg.lib`, an implementation in Singular (see http://www.singular.uni-kl.de) of the algorithms given in [Hausen et al. 2017] to compute $\mathrm{Aut}_K(R)$. Section 2 describes the algorithms and explains their implementation through examples. Section 3 is devoted to the application of our algorithms

MSC2010: 13A02, 13P10, 14J50, 14L30, 14Q15, 13A50.
Keywords: graded algebras, automorphisms, symmetries, Cox rings, Mori dream spaces, computing, Singular.
autgradalg.lib version 4.1.1.0

to *Mori dream spaces*; we determine in Proposition 3.1 information on the automorphism groups of a class of Fano threefolds listed in [Bechtold et al. 2016]. The software is available in the online supplement or at [Keicher 2017].

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 $\mathbb{Z}^k \oplus \mathbb{Z}/a_1\mathbb{Z} \oplus \cdots \oplus \mathbb{Z}/a_l\mathbb{Z}$. 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 terms of generators and relations:

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

As one can remove linear equations, it is no restriction to assume that R is *minimally presented*, i.e., $I \subseteq \langle T_1, \ldots, T_r \rangle^2$ 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:

- The homogeneous components $I_{q_1}, \ldots, I_{q_r}$ are all trivial.
- The set $\{q_1^0, \ldots, q_r^0\} \subseteq \mathbb{Z}^k$ 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` I). Let $K := \mathbb{Z}^3 \oplus \mathbb{Z}/2\mathbb{Z}$. In [Hausen and Keicher 2015, Example 2.1] and [Keicher 2014] we considered this K-graded $\mathbb{C}$-algebra R:

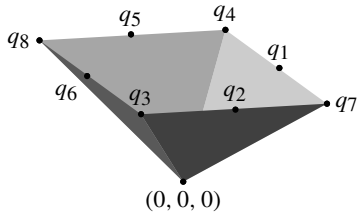
$$R \;=\; S/I, \quad S \;:=\; \mathbb{C}[T_1, \ldots, T_8], \quad I \;:=\; \langle T_1T_6 + T_2T_5 + T_3T_4 + T_7T_8 \rangle,$$

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

Then the K-grading given by Q is effective and pointed, as suggested in the picture.

To use `autgradalg.lib`, download it from the online supplement, unpack it and start Singular in the same directory. We enter R with the commands

```
> LIB "autgradalg.lib";
> intmat Q[4][8] =
>  1,1,0,0,-1,-1,2,-2,
>  0,1,1,-1,-1,0,1,-1,
>  1,1,1,1,1,1,1,1,
>  1,0,1,0,1,0,1,0;
> list TOR = 2; // torsion part of K
```



```
> ring S = 0,T(1..8),dp;
> setBaseMultigrading(Q); // grading
```

Let us recall briefly the steps of the algorithm to compute $\mathrm{Aut}_K(R)$; for details, we refer to [Hausen et al. 2017]. The overall idea is to present $\mathrm{Aut}_K(R)$ as a stabilizer in the automorphism group $\mathrm{Aut}_K(S)$ of the K-graded polynomial ring S. In a first step, we will compute a presentation $\mathrm{Aut}_K(S) \subseteq \mathrm{GL}(n)$ 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 $\mathrm{GL}(n)$-action.

Construction 2.2 [Hausen et al. 2017, Construction 3.3]. Write $\Omega_S = \{w_1, \ldots, w_s\}$ for the duplicate-free set of all q_i. Determine a $\mathbb{C}$-vector space basis $\mathcal{B}_i$ for S_{w_i} consisting of monomials. Then the concatenation $\mathcal{B} := (\mathcal{B}_1, \ldots, \mathcal{B}_s)$ is a basis for $V = \bigoplus_i S_{w_i}$. With $n := |\mathcal{B}|$, in terms of $\mathcal{B}$, each $A \in \mathrm{GL}(n)$ defines a linear map $\varphi_A : V \to V$. We obtain an algebraic action

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

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

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

It can be computed by tracking a lattice basis among the set of free parts q_i^0 of the q_i; see [Hausen et al. 2017, Remark 3.1].

Algorithm 2.3 (computing $\mathrm{Aut}_K(S)$). See [Hausen et al. 2017, Algorithm 3.7].

Input: the K-graded polynomial ring S.

- Determine $\Omega_S = \{w_1, \ldots, w_s\}$. Compute a basis $\mathcal{B}$ as in Construction 2.2.
- Define the polynomial ring $S' := \mathbb{C}[Y_{ij};\ 1 \leq i, j \leq n]$.
- Compute an ideal $J \subseteq S'$ whose equations ensure the multiplicative condition $A \cdot (f_1 f_2) = (A \cdot f_1)(A \cdot f_2)$, where $f_i \in S$, for each $A \in V(J) \subseteq \mathrm{GL}(n)$.
- Compute $\mathrm{Aut}(\Omega_S) \subseteq \mathrm{Aut}(K)$. Determine the subset $\Gamma_0 \subseteq \mathrm{Aut}(\Omega_S)$ of those B that map $\mathcal{B}_i$ bijectively to $\mathcal{B}_j$, where $w_j = B \cdot w_i$.
- For each $B \in \Gamma_0$,
  - compute an ideal $J_B \subseteq S'$ ensuring that each matrix in $V(J_B) \subseteq \mathrm{GL}(n)$ maps the component S_w to the component $S_{B \cdot w}$ where $w \in \Omega_S$, and
  - redefine $J := J \cdot J_B$.

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

Remark 2.4. (i) The third step of Algorithm 2.3 is finite; Definition 3.4(i) of [Hausen et al. 2017] for details.

(ii) The ring S' in Algorithm 2.3 is K-graded by defining $\deg(Y_{ij})$ as the degree of the i-th element of $\mathcal{B}$.

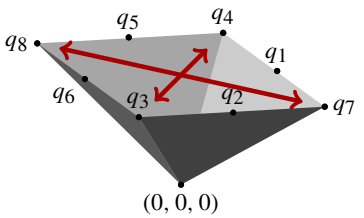
(iii) The isomorphism $S \to S$ given by $A = (a_{ij}) \in V(J) \subseteq \mathrm{GL}(n)$ is as in Construction 2.2; explicitly, it is given by $T_i \mapsto \sum_j a_{ij}(\mathcal{B}_i)_j$.

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 $\mathrm{Aut}(\Omega_R)$. In our implementation one can also trigger this step manually if desired:

```
> list origs = autGenWeights(Q, TOR);
```

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

$$\mathrm{Aut}(\Omega_S) = \left\{ \mathrm{id}, \begin{bmatrix} 1 & -2 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}, \begin{bmatrix} -1 & 2 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}, \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \right\}. \quad (1)$$



Note that $\mathrm{Aut}(\Omega_R)$ 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 $\mathrm{Aut}_K(S)$ with the command

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

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

```
> print(autKSexported[2][1]);
```

$$\begin{pmatrix}
Y(1) & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & Y(13) & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & Y(24) \\
0 & 0 & 0 & 0 & 0 & 0 & Y(31) & 0 \\
0 & Y(34) & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & Y(46) & 0 & 0 \\
0 & 0 & 0 & Y(52) & 0 & 0 & 0 & 0 \\
0 & 0 & Y(59) & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

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

```
> print(autKSexported[2][3]);
```

$$Y(2),\ Y(3),\ \ldots,\ Y(63),\ Y(64),\ -Y(1)Y(13)Y(24)Y(31)Y(34)Y(46)Y(52)Y(59)Z - 1$$

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

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

$$\mathrm{Stab}_I(\mathrm{Aut}_K(S)) \ := \ \{A \in \mathrm{Aut}_K(S);\ A \cdot I \ = \ I\} \ \subseteq \ \mathrm{Aut}_K(S).$$

Provided $I_w = \{0\}$ holds for all $w \in \Omega_S$, Hausen et al. [2017] have shown that we have an isomorphism

$$\mathrm{Stab}_I(\mathrm{Aut}_K(S)) \ \cong \ \mathrm{Aut}_K(R).$$

The final step then is the following. Define the set $\Omega_I := \{\deg(g_1), \ldots, \deg(g_s)\}$ 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 $\mathrm{Aut}_K(R)$). See [Hausen et al. 2017, Algorithm 3.8].

Input: the K-graded polynomial ring S and the defining ideal $I \subseteq S$ of R.

- Let $J \subseteq S' := \mathbb{C}[Y_{ij};\ 1 \le i, j \le n]$ be the output of Algorithm 2.3.
- Compute Ω_I and form the $\mathbb{C}$-vector space $W := \bigoplus_{\Omega_I} S_u$.
- 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 $\mathrm{GL}(n)$-action from Construction 2.2 and $Y = (Y_{ij})$, we obtain the ideal

$$J' \ := \ \langle \ell_i(Y \cdot h_j);\ 1 \le i \le m,\ 1 \le j \le l \rangle \ \subseteq \ S'.$$

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

Remark 2.7. (i) Algorithms 2.3 and 2.6 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 $G := \mathrm{Aut}_K(R) \subseteq \mathrm{GL}(n)$ 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 $\mathrm{Aut}_K(R)$. We first switch back to S, enter the defining ideal I for $R = S/I$ and start the computation of $\mathrm{Aut}_K(R)$:

```
> 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 `Sres` is identical to `Sprime`. A list `stabExported` is being exported; the interpretation of the entries is identical to that of the list `listAutKS` from Example 2.5, with the difference that the ideal part now contains additional equations describing the stabilizer: for example

```
> stabExported[2][3];
```

$$Y(2),\ Y(3),\ \ldots Y(63),\ Y(64),\ -Y(1)Y(13)Y(24)Y(31)Y(34)Y(46)Y(52)Y(59)Z - 1,$$
$$-Y(24)Y(31) + Y(52)Y(59),\ Y(13)Y(34) - Y(52)Y(59),\ -Y(13)Y(34) + Y(1)Y(46)$$

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

3. APPLICATION: MORI DREAM SPACES. In this section, we briefly recall from [Hausen et al. 2017] 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 $\mathrm{Cl}(X)$ one can assign a $\mathrm{Cl}(X)$-graded $\mathbb{C}$-algebra, its so-called *Cox ring*,

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

see, e.g., [Arzhantsev et al. 2015] 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 [Cox 1995; Birkar et al. 2010]. The Cox ring has strong implications on the underlying Mori dream space. More precisely,

X can be recovered as a good quotient

$$\operatorname{Spec}(R) =: \overline{X} \supseteq \widehat{X} \xrightarrow{/\!\!/ H} X \tag{2}$$

of an open subset $\widehat{X}$ by the *characteristic quasitorus* $H := \operatorname{Spec}(\mathbb{C}[K])$. In fact, $\widehat{X}$ is determined by an ample class $w \in \operatorname{Cl}(X)$. This opens up a computer algebra based approach [Hausen and Keicher 2015; Keicher 2014] to Mori dream spaces. In [Arzhantsev et al. 2014], it has been shown that (2) translates to automorphisms of X as follows:

$$\operatorname{Aut}_{\operatorname{Cl}(X)}(\operatorname{Cox}(X)) \cong \operatorname{Aut}_H(\overline{X}) \supseteq \operatorname{Aut}_H(\widehat{X}) \xrightarrow{/H} \operatorname{Aut}(X) \tag{3}$$

Here, by $\operatorname{Aut}_H(Y)$ we mean the group of *H-equivariant automorphisms* of Y; these are pairs (φ, ψ) with $\varphi : Y \to Y$ being an automorphism of varieties and $\psi : H \to H$ 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 can directly compute $\operatorname{Aut}_H(\overline{X})$ with Algorithm 2.6. In the following proposition, we investigate the symmetries of the list of Fano varieties [Bechtold et al. 2016].

Proposition 3.1. *Let X_i be the nontoric terminal Fano threefold of Picard number one with an effective two-torus action from the classification in* [*Bechtold et al. 2016, Theorem* 1.1].

(i) *For all $1 \le i \le 41$, Algorithm 2.6 is able to compute a presentation of $G_i := \operatorname{Aut}_H(\overline{X}_i)$ as an affine algebraic subgroup $V(J_i) \subseteq \operatorname{GL}(n_i)$.*

(ii) *Using* (i), *these are the dimensions* $\dim(G_i)$ *and the number of components* $[G_i : G_i^0]$ *of a selection of $G_i \subseteq \operatorname{GL}(n_i)$*:

X_i	$\operatorname{Aut}(\mathfrak{Q}_S)$	$\dim G_i$	$[G_i : G_i^0]$	$\dim \operatorname{Aut}(X_i)$	X_i	$\operatorname{Aut}(\mathfrak{Q}_S)$	$\dim G_i$	$[G_i : G_i^0]$	$\dim \operatorname{Aut}(X_i)$
X_3	$\mathbb{Z}/4\mathbb{Z}$	3	4	2	X_{26}	$\mathbb{Z}/2\mathbb{Z}$	3		2
X_6	$\{1\}$	5		4	X_{28}	$\{1\}$	4	1	3
X_7	$\{1\}$	5		4	X_{33}	$\{1\}$	6	2	5
X_{10}	$\{1\}$	4	1	3	X_{34}	$\{1\}$	6	2	5
X_{12}	$\{1\}$	6		5	X_{36}	$\{1\}$	5	1	4
X_{13}	$\{1\}$	4	1	3	X_{37}	$\{1\}$	4	2	3
X_{14}	$\{1\}$	3		2	X_{38}	$\{1\}$	4	3	3
X_{15}	$\{1\}$	5		4	X_{39}	$\{1\}$	3		2
X_{16}	$\{1\}$	3		2	X_{40}	$\{1\}$	3	1	2
X_{18}	$\{1\}$	6		5	X_{42}	$\{1\}$	3	2	2
X_{19}	$\{1\}$	4	1	3	X_{45}	$\{1\}$	4	2	3
X_{20}	$\{1\}$	5		4	X_{46}	$\{1\}$	4	1	3
X_{21}	$\{1\}$	3	2	2	X_{47}	$\{1\}$	3	1	2
X_{25}	$\{1\}$	4	1	3					

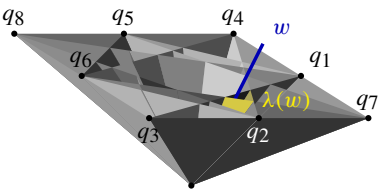
Proof. This is an application of Algorithm 2.6 and of the Singular commands to compute dimension and absolute components; see, for example, [Greuel and Pfister 2008]. We performed the computations on an older machine (Intel celeron CPU, 4 GB RAM) and canceled them after several seconds. The files are available at [Keicher 2017]. □

In [Hausen et al. 2017], the authors have also presented algorithms to compute $\mathrm{Aut}_H(\widehat{X})$ and generators for the Hopf algebra $\mathcal{O}(\mathrm{Aut}(X))$. Both algorithms are also implemented in our library. However, the case $\mathcal{O}(\mathrm{Aut}(X))$ 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 Example 2.8, we have already computed $\mathrm{Aut}_H(\overline{X}) \cong G := \mathrm{Aut}_K(R)$. From it, we obtain $\mathrm{Aut}_H(\widehat{X})$ as follows: first, w defines a certain polyhedral cone, the GIT-*cone* $\lambda(w)$. Then $\mathrm{Aut}_H(\widehat{X})$ is obtained from G by choosing only those elements (A_B, B, J_B) of the list `stabExported` where $B \in \mathrm{Aut}(\Omega_S)$ fixes $\lambda(w)$. In our library, you can compute it as follows (making use of `gitfan.lib` [Boehm et al. 2016]):

```
> intvec w = 1,9,16,0; // drawn in blue
> setring S; // from before, R=S/I
> def RR = autXhat(I, w, TOR);
> setring RR;
```



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

SUPPLEMENT. The online supplement contains version 4.1.1.0 of autgradalg.lib.

REFERENCES.

[Arzhantsev et al. 2014] 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**:3 (2014), 429–471, 641. MR Zbl

[Arzhantsev et al. 2015] I. Arzhantsev, U. Derenthal, J. Hausen, and A. Laface, *Cox rings*, Cambridge Studies in Advanced Mathematics **144**, Cambridge University Press, 2015. MR Zbl

[Bechtold et al. 2016] B. Bechtold, J. Hausen, E. Huggenberger, and M. Nicolussi, "On terminal Fano 3-folds with 2-torus action", *Int. Math. Res. Not.* **2016**:5 (2016), 1563–1602. MR Zbl

[Birkar et al. 2010] C. Birkar, P. Cascini, C. D. Hacon, and J. McKernan, "Existence of minimal models for varieties of log general type", *J. Amer. Math. Soc.* **23**:2 (2010), 405–468. MR Zbl

[Boehm et al. 2016] J. Boehm, S. Keicher, and Y. Ren, "Computing GIT-fans with symmetry and the Mori chamber decomposition of $\bar{M}_{0,6}$", 2016. arXiv

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

[Greuel and Pfister 2008] G.-M. Greuel and G. Pfister, *A* SINGULAR *introduction to commutative algebra*, 2nd ed., Springer, 2008. MR

[Hausen and Keicher 2015] J. Hausen and S. Keicher, "A software package for Mori dream spaces", *LMS J. Comput. Math.* **18**:1 (2015), 647–659. MR Zbl

[Hausen et al. 2017] J. Hausen, S. Keicher, and R. Wolf, "Computing automorphisms of Mori dream spaces", *Math. Comp.* **86**:308 (2017), 2955–2974. MR Zbl

[Jensen 2017] A. N. Jensen, "Gfan, a software system for Gröbner fans and tropical varieties", 2017, http://home.imf.au.dk/jensen/software/gfan/gfan.html. Zbl

[Keicher 2014] S. Keicher, *Algorithms for Mori dream spaces*, Ph.D. thesis, Universität Tübingen, 2014, https://publikationen.uni-tuebingen.de/xmlui/handle/10900/54061. Zbl

[Keicher 2017] S. Keicher, "`autgradalg.lib` – a library for `Singular` to compute automorphisms of graded algebras", 2017, https://github.com/skeicher/autgradalg-lib.

[Steidel 2013] S. Steidel, "Gröbner bases of symmetric ideals", *J. Symbolic Comput.* **54** (2013), 72–86. MR Zbl

RECEIVED: 16 Apr 2017 REVISED: 18 Apr 2018 ACCEPTED: 18 May 2018

SIMON KEICHER:

keicher@mail.mathematik.uni-tuebingen.de
Mathematisches Institut, Universität Tübingen, Tübingen, Germany

JOURNAL OF SOFTWARE FOR ALGEBRA AND GEOMETRY vol 8, no 1, 2018

HeLP: a GAP package for torsion units in integral group rings 1
Andreas Bächle and Leo Margolis

A software package to compute automorphisms of graded algebras 11
Simon Keicher

A package for computations with classical resultants 21
Giovanni Staglianò

The SpaceCurves package in Macaulay2 31
Mengyuan Zhang

The ReesAlgebra package in Macaulay2 49
David Eisenbud

A Macaulay2 package for computations with rational maps 61
Giovanni Staglianò

ExteriorIdeals: a package for computing monomial ideals in an exterior algebra 71
Luca Amata and Marilena Crupi

Software for computing conformal block divisors on $\overline{M}_{0,n}$ 81
David Swinarski

Divisor Package for Macaulay2 87
Karl Schwede and Zhaoning Yang

1948-7916(2018)8:1;1-F