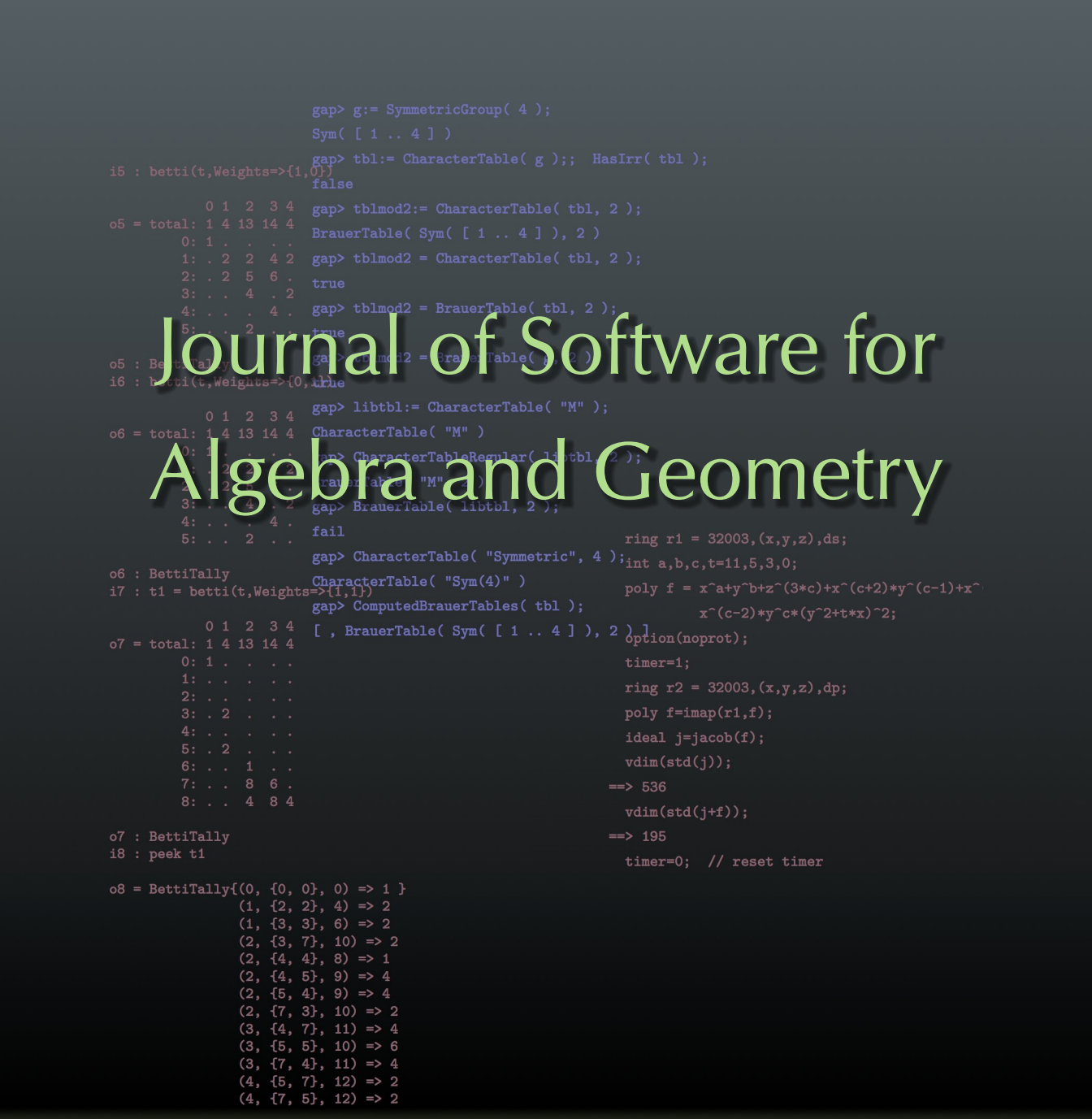


Journal of Software for Algebra and Geometry

vol 8 2018

JSAG **8** (2018), 1–9
dx.doi.org/10.2140/jsag.2018.8.1

The Journal of Software for
Algebra and Geometry

HeLP: a GAP package for torsion units in integral group rings

ANDREAS BÄCHLE AND LEO MARGOLIS

ABSTRACT: We briefly summarize the background of the HeLP method for torsion units in group rings and present some functionality of a GAP package implementing it.

1. THE ZASSENHAUS CONJECTURE AND THE PRIME GRAPH QUESTION. As we consider the integral group ring $\mathbb{Z}G$ of a finite group G, one question that arises naturally is: "What does the unit group $\mathrm{U}(\mathbb{Z}G)$ look like?" For example, what are the *torsion units*, i.e., the units of finite order? Clearly, there are the so-called *trivial units* $\pm g$ for $g \in G$. Already in G. Higman's PhD thesis [1940], it was proved that all the torsion units are of this form, provided G is abelian. As we are not interested in the torsion coming solely from the ring, but rather in the torsion coming from the group-ring interplay, we consider the group of *normalized units* $\mathrm{V}(\mathbb{Z}G)$, i.e., the units mapping to 1 under the augmentation homomorphism

$$\varepsilon : \mathbb{Z}G \to \mathbb{Z} : \sum_{g\in G} u_g g \mapsto \sum_{g\in G} u_g.$$

Then $\mathrm{U}(\mathbb{Z}G) = \pm\mathrm{V}(\mathbb{Z}G)$.

In the noncommutative case, there are in general, of course, more torsion units than the trivial ones, e.g., conjugates of group elements by units of $\mathbb{Q}G$ which end up in $\mathbb{Z}G$ again. H. J. Zassenhaus conjectured more than 40 years ago that these are all the torsion units.

Zassenhaus conjecture (ZC) [1974]. *Let G be a finite group and u a torsion unit in $\mathrm{V}(\mathbb{Z}G)$. Then there exists a unit x in $\mathbb{Q}G$ such that $x^{-1}ux = g$ for some $g \in G$.*

Elements $u, v \in \mathbb{Z}G$ which are conjugate by a unit $x \in \mathbb{Q}G$ are called *rationally conjugate*, denoted by $u \sim_{\mathbb{Q}G} v$. The Zassenhaus conjecture is nowadays one of the

Bächle was supported by the Research Foundation Flanders (FWO - Vlaanderen). This project was partly supported by the DFG priority program SPP 1489.

MSC2010: primary 16U60, 16Z05; secondary 16S34, 20C05.

Keywords: integral group ring, torsion units, Zassenhaus conjecture, prime graph question, computer algebra, GAP.

main open questions in the area of integral group rings. A highlight was certainly Weiss' proof of this conjecture for nilpotent groups [Weiss 1991].

As a first step towards the Zassenhaus conjecture, W. Kimmerle formulated a weaker version which has attracted attention.

Prime graph question (PQ) [Kimmerle 2006]. Let G be a finite group and p and q different primes such that $\mathrm{V}(\mathbb{Z}G)$ contains an element of order pq. Does G then possess an element of order pq?

Given a group G the prime graph of G is defined to be the graph whose vertices are labeled by primes appearing as orders of elements in G, and two vertices p and q are connected by an edge if and only if G contains an element of order pq. Thus (PQ) asks whether G and $\mathrm{V}(\mathbb{Z}G)$ have the same prime graph.

A method to attack these questions, known as the HeLP method, was introduced by Luthar and Passi [1989] and later extended by Hertweck [2007]. The name *HeLP* (*He*rtweck *L*uthar *P*assi) is due to A. Konovalov. The method can be applied algorithmically to a concrete group or, if one has generic characters at hand, a series of groups.

The present note presents a GAP package implementing this method [HeLP package]. The main motivation for this program is to make the algorithm available to researchers working in the field, and to enable readers of papers using the method to check results obtained by the method. We describe the method in Section 2 and discuss several aspects of our implementation in Sections 3 and 4.

2. The HeLP constraints.

Let G always be a finite group.

The possible orders of torsion units in $\mathbb{Z}G$ are restricted:

Proposition 2.1. *Let $u \in \mathrm{V}(\mathbb{Z}G)$ be a torsion unit.*

(a) *The order of u divides the exponent of G* [Cohn and Livingstone 1965, Corollary 4.1].

(b) *If G is solvable, then the order of u coincides with the order of an element of G* [Hertweck 2008, Theorem].

Definition 2.2. Let $u = \sum_{g\in G} u_g g \in \mathbb{Z}G$, $x \in G$ and denote by x^G its conjugacy class. Then

$$\varepsilon_x(u) = \sum_{g\in x^G} u_g$$

is called the *partial augmentation of u with respect to x* (or, rather, the conjugacy class of x).

The following proposition by Marciniak, Ritter, Sehgal and Weiss connects (ZC) to partial augmentations.

Proposition 2.3 [Marciniak et al. 1987, Theorem 2.5]. *Let $u \in \mathrm{V}(\mathbb{Z}G)$ be a torsion unit of order k. Then u is rationally conjugate to a group element if and only if $\varepsilon_x(u^d) \geq 0$ for all divisors d of k and all $x \in G$.*

Certain partial augmentations vanish a priori:

Proposition 2.4. *Let $u \in \mathrm{V}(\mathbb{Z}G)$ be a torsion unit and $x \in G$.*

(a) *If $o(u) \neq 1$, then $\varepsilon_1(u) = 0$* (Berman and Higman; see [Sehgal 1993, Proposition 1.4]).

(b) *If $o(x) \nmid o(u)$, then $\varepsilon_x(u) = 0$* [Hertweck 2007, Theorem 2.3].

Let ψ be a character of the group G. A representation afforded by ψ can be extended linearly to a representation of $\mathbb{Q}G$ and then restricted to a representation D of the group of units $\mathrm{U}(\mathbb{Q}G)$. We will denote its character also by ψ. Now consider for a torsion unit $u \in \mathrm{V}(\mathbb{Z}G)$ of order k, a linear character $\chi : C_k \simeq \langle u \rangle \to \mathbb{C}$ given by $\chi(u) = \zeta^\ell$, with $\zeta \in \mathbb{C}^\times$ a primitive k-th root of unity, $\ell \in \mathbb{Z}$. Then we have that the multiplicity of ζ^ℓ as an eigenvalue of $D(u)$ is given by

$$\langle \chi, \psi \rangle_{\langle u \rangle} \in \mathbb{Z}_{\geq 0},$$

where $\langle -, - \rangle_{\langle u \rangle}$ denotes the inner product on the class functions of $C_k \simeq \langle u \rangle$. Working out an explicit formula for this, one obtains part (a) of the following proposition.

Proposition 2.5. *Let G be a finite group and $u \in \mathrm{V}(\mathbb{Z}G)$ a torsion unit of order k. Let $\zeta \in \mathbb{C}^\times$ be a primitive k-th root of unity, $\ell \in \mathbb{Z}$.*

(a) [Luthar and Passi 1989, Theorem 1] *Let χ be an ordinary character of G and let D be a representation afforded by χ. Then the multiplicity of ζ^ℓ as an eigenvalue of $D(u)$ is given by*

$$\mu_\ell(u, \chi) = \frac{1}{k} \sum_{d \mid k} \mathrm{Tr}_{\mathbb{Q}(\zeta^d)/\mathbb{Q}}(\chi(u^d)\zeta^{-d\ell}). \tag{1}$$

(b) [Hertweck 2007, Section 4] *Let p be a prime not dividing k, and $\zeta \mapsto \bar{\zeta}$ be a fixed isomorphism between the group of k-th roots of unity in characteristic 0 and those in characteristic p. Let φ be a p-Brauer character of G and P be a representation afforded by φ. Then the multiplicity of $\bar{\zeta}^\ell$ as an eigenvalue of $P(u)$ is given by*

$$\mu_\ell(u, \varphi) = \frac{1}{k} \sum_{d \mid k} \mathrm{Tr}_{\mathbb{Q}(\zeta^d)/\mathbb{Q}}(\varphi(u^d)\zeta^{-d\ell}). \tag{2}$$

This proposition is the linchpin of the HeLP method. Let $u \in \mathrm{V}(\mathbb{Z}G)$ be again a torsion unit of order k. For an ordinary character χ we have $\chi(u) = \sum_{x^G} \varepsilon_x(u)\chi(x)$. By [Hertweck 2007, Theorem 3.2], an analogous statement holds for p-Brauer

characters and p-regular units u, where the sum is taken only over the conjugacy classes of p-regular elements in G (an element is called *p-regular* if its order is not divisible by p). Assume for an ordinary or a Brauer character ψ in characteristic $p\nmid k$ one knows inductively the character values of $\psi(u^d)$. Then the following condition on the $\varepsilon_x(u)$ holds for every ℓ:

$$\sum_{x^G} \frac{\mathrm{Tr}_{\mathbb{Q}(\zeta)/\mathbb{Q}}(\psi(x)\zeta^{-\ell})}{k}\varepsilon_x(u) + a_\ell(u,\psi) \in \mathbb{Z}_{\geq 0}, \tag{3}$$

where the $a_\ell(u,\psi) = \frac{1}{k}\sum_{1\neq d\,|\,k} \mathrm{Tr}_{\mathbb{Q}(\zeta^d)/\mathbb{Q}}(\psi(u^d)\zeta^{-d\ell})$ are assumed to be "known". By Proposition 2.4 it is enough to take the sum (3) over classes of elements having an order dividing k.

Remarks 2.6.

- By [Hales et al. 1990, Corollary 2.3], the partial augmentations are bounded and thus solving the inequalities is a finite problem. These bounds are, however, encoded in the ordinary character table, so they will not add new information to the algorithm.
- It is intrinsic in the formula that the μ_ℓ's sum up to the degree of the character.

3. THE EXTENDED WAGNER TEST. The program uses also a criterion proved in a special form by Roland Wagner in his Diplomarbeit [1995]. Proposition 3.1, the more general case, is recorded in [Bovdi and Hertweck 2008, Remark 6] and follows from [Sehgal 1993, Lemma 7.1], which is well known. For the sake of completeness, we include a proof since no short complete proof seems to be available in the literature. We write $g \sim h$ if g and h are conjugate in a group G.

Proposition 3.1. *Let G be a finite group, $s \in G$ and $u \in \mathrm{V}(\mathbb{Z}G)$. Let p be a prime and j a nonnegative integer. Then*

$$\sum_{x^G,\ x^{p^j}\sim s} \varepsilon_x(u) \equiv \varepsilon_s(u^{p^j}) \mod p.$$

Proof. Let $u = \sum_{g\in G} u_g g \in \mathrm{V}(\mathbb{Z}G)$, set $q = p^j$ and $v = u^q$. By definition,

$$\varepsilon_s(v) = \sum_{\substack{(g_1,\dots,g_q)\in G^q \\ g_1\cdots g_q \sim s}} \prod_{j=1}^{q} u_{g_j}. \tag{4}$$

The set over which the sum is taken can be decomposed into $\mathcal{M} = \{(g,\dots,g) \in G^q : g^q \sim s\}$ and $\mathcal{N} = \{(g_1,\dots,g_q) \in G^q : g_1\cdots g_q \sim s \text{ and there exists } r, r' : g_r \neq g_{r'}\}$.

The cyclic group $C_q = \langle t\rangle$ of order q acts on the set $\mathcal{N}$ by letting the generator t shift the entries of a tuple to the left, i.e.,

$$(g_1, g_2, g_3, \dots, g_q)\cdot t = (g_2, g_3, \dots, g_q, g_1).$$

Note that all orbits have length p^i with $i \geq 1$. For elements in the same orbit, the same integer is summed up in (4). Hence using Fermat's little theorem we have

$$\begin{aligned}\varepsilon_s(v) &= \sum_{(g,\dots,g)\in\mathcal{M}} u_g^q + \sum_{(g_1,\dots,g_q)\in\mathcal{N}} \prod_{j=1}^{q} u_{g_j} \\ &\equiv \sum_{(g,\dots,g)\in\mathcal{M}} u_g^q \equiv \sum_{(g,\dots,g)\in\mathcal{M}} u_g \equiv \sum_{x^G,\ x^{p^j}\sim s} \varepsilon_x(u) \mod p. \qquad \square\end{aligned}$$

By induction and the Berman–Higman result (Proposition 2.4 (a)), Wagner obtained the following. For units of prime power order the result also follows from [Cohn and Livingstone 1965, Theorem 4.1].

Corollary 3.2 (Wagner). *Let G be a finite group, $u \in \mathrm{V}(\mathbb{Z}G)$, $o(u) = p^j m$ with p a prime and $m \neq 1$. Then*

$$\sum_{x^G,\ o(x)=p^j} \varepsilon_x(u) \equiv 0 \mod p.$$

Example 3.3. Let G be the Mathieu group of degree 11. There exists only one conjugacy class of involutions in G, call it `2a`. After applying HeLP (i.e., Proposition 2.5) for a unit u of order 12 in $\mathrm{V}(\mathbb{Z}G)$, one obtains two possible partial augmentations for u. One of these possibilities satisfies $\varepsilon_{\mathtt{2a}}(u) = 1$ while the other satisfies $\varepsilon_{\mathtt{2a}}(u) = -1$ [Bovdi and Konovalov 2007]. Neither possibility satisfies the constraints of Wagner's result and thus there are no torsion units of order 12 in $\mathrm{V}(\mathbb{Z}G)$ and the order of any torsion unit in $\mathrm{V}(\mathbb{Z}G)$ coincides with the order of an element in G.

4. Implementation.

4A. ***Further results.*** We used several results in our implementation, which are not consequences of the HeLP method and which we list here. The first one is a direct consequence of the Fong–Swan–Rukolaine theorem [Curtis and Reiner 1981, Theorem 22.1].

Proposition 4.1. *Let G be a p-solvable group and $u \in \mathrm{V}(\mathbb{Z}G)$ a torsion unit of order prime to p. Then the restrictions on the possible partial augmentations of u one can obtain using the p-Brauer table of G are the same as when using the ordinary character table of G.*

To avoid redundant calculations, our implementation also uses the following results instead of solving any inequalities in these situations.

Proposition 4.2. (a) *(ZC) holds for nilpotent groups* [Weiss 1991].

(b) *(PQ) has an affirmative answer for solvable groups* [Kimmerle 2006].

Remark 4.3. There are other results about (ZC) and (PQ). For example, (ZC) is known for cyclic-by-abelian groups [Caicedo et al. 2013], while (PQ) is known for PSL(2, p) where p denotes a prime [Hertweck 2007]. However, we decided for simplicity only to use the results in Proposition 4.2 in the package.

4B. ***Main functions of the HeLP package.*** The function `HeLP_ZC` checks whether (ZC) can be verified using the character tables and Brauer tables available in GAP. For a potential element u of order k whose partial augmentations we want to compute, and p and q different prime divisors of k, we call partial augmentations of u^p and u^q *compatible* if $(u^p)^q$ and $(u^q)^p$ have the same partial augmentations.

Algorithm 1: HeLP_ZC

Input: group or ordinary character table of a group
Output: `true` or `false`
if *G nilpotent* **then**
  return `true` (see Proposition 4.2)
end
if *G solvable* **then**
  `OrdersToCheck :=` orders of elements in G (see Proposition 2.1)
else
  `OrdersToCheck :=` divisors of $\exp G$
end
for $k = o(u)$ *in* `OrdersToCheck` **do**
  for *all prime divisors p of* $o(u)$ *and all possible partial augmentations of* u^p **do**
    if *partial augmentations are compatible* **then**
      Construct and solve the HeLP systems for all relevant character tables
    end
  end
  Apply the Wagner test for order k
  Save the resulting possibilities for partial augmentations of units of order k in the global variable `HeLP_sol`
end
if *only "trivial" partial augmentations are admissible* **then return** `true`
else return `false`

The function `HeLP_PQ` checks whether (PQ) can be verified using the character tables and Brauer tables available in GAP. It works in a similar way to `HeLP_ZC` but only checks the orders relevant for the prime graph question.

The package contains other functions, such as one that allows the user to check whether units of a given order occur; for further details, see the reference manual.

4C. ***Nonstandard characters.*** Unfortunately, not all known character tables and Brauer tables are available in GAP, so the package cannot be applied in those cases. However, there are some workarounds.

Example 4.4. The Brauer table modulo 7 of PSL(2, 49) is known generically, but not yet included in the GAP Character Table Library [CTblLib]. However, our implementation allows the use of class functions of a group provided by the user. In this way, any class function may be used and it is, among other things, possible to prove (ZC) for PSL(2, 49).

Example 4.5. Let G be the projective unitary group PSU(3, 8) and A its automorphism group. Assume the goal is to check (PQ) for A. To obtain that one needs to exclude the existence of units of order $2 \cdot 19$ and $7 \cdot 19$ in $\mathrm{V}(\mathbb{Z}A)$. The character table of G is available in GAP while that of A is not. However, inducing the second and third irreducible characters of G to characters of A, one obtains two characters of A. The HeLP constraints following from these two characters are strong enough to prove (PQ) for A.

4D. ***Solving the inequalities.*** Applying the HeLP method involves solving the integral linear inequalities described after Proposition 2.5. This is a hard task in general: although theoretically possible, it may take a lot of time when there are many inequalities and variables involved. A good solver of such systems is the main requirement here. Our implementation allows the use of two solvers; the software system [4ti2, version ≥1.6.5], and/or the system [Normaliz, version ≥3.1.0] for rational cones and affine models (see also [Bruns et al. 2016]). We chose those solvers because they are good solvers and there exist GAP interfaces for them [4ti2Interface; NormalizInterface]. To reduce the size of the system that must be solved, the package uses the algorithm "redund" from [lrslib, version ≥4.3] for reverse-search vertex enumeration. When using 4ti2, in many cases this leads to a remarkable speedup; however, it may slow down the calculations, so there is an option implemented to switch the use of "redund" on and off.

4E. ***p-constant characters.*** If one is interested especially in solving (PQ) there is often a way to reduce the system one has to solve, which was introduced by V. Bovdi and A. Konovalov [2010]. Assume one is studying the possible partial augmentations of units of order $p \cdot q$, where p and q are different primes. Let χ be a character which is constant on all conjugacy classes of elements of order p, a so called *p-constant character*. Then the coefficients appearing in the HeLP constraints provided by χ at partial augmentations of elements order p are always the same. Thus one can reduce the number of variables involved by replacing all the partial augmentations of elements of order p by their sum. This way one also does not need to know the partial augmentations of elements of order p — their sum

is 1 in any case. Often it suffices to study only p-constant characters to exclude the possibility of existence of units of order $p \cdot q$ and this functionality is also provided by the package.

The package is available as an online supplement.

REFERENCES.

[4ti2] 4ti2 team, "4ti2 — a software package for algebraic, geometric and combinatorial problems on linear spaces", available at www.4ti2.de.

[4ti2Interface] S. Gutsche, "4ti2 interface", GAP package, version 2015.11.06, available at https://tinyurl.com/4ti2gapInterface.

[Bovdi and Hertweck 2008] V. Bovdi and M. Hertweck, "Zassenhaus conjecture for central extensions of S_5", *J. Group Theory* **11**:1 (2008), 63–74. MR Zbl

[Bovdi and Konovalov 2007] V. Bovdi and A. Konovalov, "Integral group ring of the first Mathieu simple group", pp. 237–245 in *Groups St. Andrews 2005*, vol. 1, edited by C. M. Campbell et al., London Math. Soc. Lecture Note Ser. **339**, Cambridge Univ. Press, 2007. MR Zbl

[Bovdi and Konovalov 2010] V. A. Bovdi and A. B. Konovalov, "Torsion units in integral group ring of Higman–Sims simple group", *Studia Sci. Math. Hungar.* **47**:1 (2010), 1–11. MR Zbl

[Bruns et al. 2016] W. Bruns, B. Ichim, and C. Söger, "The power of pyramid decomposition in Normaliz", *J. Symbolic Comput.* **74** (2016), 513–536. MR Zbl

[Caicedo et al. 2013] M. Caicedo, L. Margolis, and A. del Río, "Zassenhaus conjecture for cyclic-by-abelian groups", *J. Lond. Math. Soc.* (2) **88**:1 (2013), 65–78. MR Zbl

[Cohn and Livingstone 1965] J. A. Cohn and D. Livingstone, "On the structure of group algebras, I", *Canad. J. Math.* **17** (1965), 583–593. MR Zbl

[CTblLib] T. Breuer, "The GAP character table library", GAP package, version 1.2.1, available at http://www.math.rwth-aachen.de/∼Thomas.Breuer/ctbllib.

[Curtis and Reiner 1981] C. W. Curtis and I. Reiner, *Methods of representation theory, I: With applications to finite groups and orders*, Wiley, New York, 1981. MR Zbl

[Hales et al. 1990] A. W. Hales, I. S. Luthar, and I. B. S. Passi, "Partial augmentations and Jordan decomposition in group rings", *Comm. Algebra* **18**:7 (1990), 2327–2341. MR Zbl

[HeLP package] A. Bächle and L. Margolis, "HeLP – Hertweck–Luthar–Passi method", GAP package, version 3.1, available at http://homepages.vub.ac.be/abachle/help/.

[Hertweck 2007] M. Hertweck, "Partial augmentations and Brauer character values of torsion units in group rings", 2007. arXiv

[Hertweck 2008] M. Hertweck, "The orders of torsion units in integral group rings of finite solvable groups", *Comm. Algebra* **36**:10 (2008), 3585–3588. MR Zbl

[Higman 1940] G. Higman, "The units of group-rings", *Proc. London Math. Soc.* (2) **46** (1940), 231–248. MR Zbl

[Kimmerle 2006] W. Kimmerle, "On the prime graph of the unit group of integral group rings of finite groups", pp. 215–228 in *Groups, rings and algebras*, edited by W. Chin et al., Contemp. Math. **420**, Amer. Math. Soc., Providence, RI, 2006. MR Zbl

[lrslib] D. Avis, "lrslib", software version 6.2, available at http://cgm.cs.mcgill.ca/∼avis/C/lrs.html.

[Luthar and Passi 1989] I. S. Luthar and I. B. S. Passi, "Zassenhaus conjecture for A_5", *Proc. Indian Acad. Sci. Math. Sci.* **99**:1 (1989), 1–5. MR Zbl

[Marciniak et al. 1987] Z. Marciniak, J. Ritter, S. K. Sehgal, and A. Weiss, "Torsion units in integral group rings of some metabelian groups, II", *J. Number Theory* **25**:3 (1987), 340–352. MR Zbl

[Normaliz] W. Burns, B. Ichim, R. Sieg, and C. Söger, "Normaliz", software, version 3.2.0, available at http://normaliz.uos.de.

[NormalizInterface] S. Gutsche, M. Horn, and C. Söger, "NormalizInterface", GAP package, version 0.9.8, available at http://gap-packages.github.io/NormalizInterface/.

[Sehgal 1993] S. K. Sehgal, *Units in integral group rings*, Pitman Monographs and Surveys in Pure and Applied Mathematics **69**, Longman, Harlow, 1993. MR Zbl

[Wagner 1995] R. Wagner, *Zassenhausvermutung über die Gruppen* PSL(2, p), Diplomarbeit, Universität Stuttgart, 1995.

[Weiss 1991] A. Weiss, "Torsion units in integral group rings", *J. Reine Angew. Math.* **415** (1991), 175–187. MR Zbl

[Zassenhaus 1974] H. Zassenhaus, "On the torsion units of finite group rings", pp. 119–126 in *Studies in mathematics* (*in honor of A. Almeida Costa*), Instituto de Alta Cultura, Lisbon, 1974. MR Zbl

RECEIVED: 1 Oct 2015 REVISED: 6 Oct 2017 ACCEPTED: 18 May 2018

ANDREAS BÄCHLE:

abachle@vub.ac.be
Vakgroep Wiskunde, Vrije Universiteit Brussel, Pleinlaan 2, 1050 Brussels, Belgium

LEO MARGOLIS:

leo.margolis@um.es
Fachbereich Mathematik, Universität Stuttgart, Pfaffenwaldring 57, 70569 Stuttgart, Germany

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:

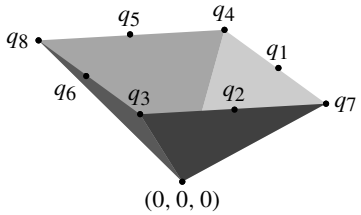
$$R \;=\; S/I, \quad S \;:=\; \mathbb{C}[T_1, \ldots, T_8], \quad I \;:=\; \langle T_1T_6 + T_2T_5 + T_3T_4 + T_7T_8 \rangle,$$

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

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 $\mathcal{B}$.

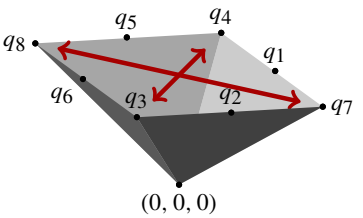
(iii) The isomorphism $S \to S$ given by $A = (a_{ij}) \in V(J) \subseteq \mathrm{GL}(n)$ is as in Construction 2.2; explicitly, it is given by $T_i \mapsto \sum_j a_{ij}(\mathcal{B}_i)_j$.

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					

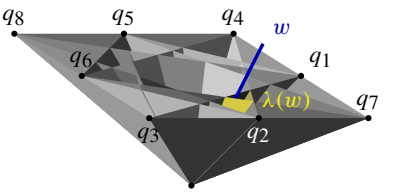
Proof. This is an application of Algorithm 2.6 and of the Singular commands to compute dimension and absolute components; see, for example, [Greuel and Pfister 2008]. We performed the computations on an older machine (Intel celeron CPU, 4 GB RAM) and canceled them after several seconds. The files are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

JSAG **8** (2018), 21–30
dx.doi.org/10.2140/jsag.2018.8.21

The Journal of Software for
Algebra and Geometry

A package for computations with classical resultants

GIOVANNI STAGLIANÒ

ABSTRACT: We present the Macaulay2 package *Resultants*, which provides commands for the effective computation of multivariate resultants, discriminants, and Chow forms. We provide some background for the algorithms implemented and show, with a few examples, how the package works.

INTRODUCTION. The *resultant* characterizes the existence of nontrivial solutions for a square system of homogeneous polynomial equations as a condition on the coefficients. One of its important features is that it can be used to compute elimination ideals and to solve polynomial equations. Indeed, it provides one of the two main tools in elimination theory, along with Gröbner bases. The resultant of the system of equations given by the partial derivatives of a complex homogeneous polynomial F is called (up to a constant factor) the *discriminant* of F. It characterizes the existence of singular points in the projective hypersurface $V(F)$ as a condition on the coefficients of F. In this special case, all polynomial equations have the same total degree. Every time the system of equations consists of $n+1$ polynomial equations of the same total degree d, the resultant has a further interesting property: it can be expressed as a polynomial of degree d^n in the $(n+1)\times(n+1)$ minors of an $(n+1)\times\binom{n+d}{n}$ matrix, the coefficient matrix of the system of equations. This allows us to write down a *generic* resultant in a more compact form. The polynomial of degree d^n so obtained is geometrically interpreted as the *Chow form* of the d-th Veronese embedding of $\mathbb{P}^n$.

The package *Resultants*, included with [Macaulay2], provides commands for the explicit computation of resultants and discriminants. The main algorithm used is based on the so-called *Poisson formula*, which reduces the computation of the resultant of $n+1$ equations to the product of the resultant of n equations with the determinant of an appropriate matrix. This algorithm requires a certain genericity condition on the input polynomials, achievable with a generic change of coordinates. The package also includes tools for working with Chow forms and more generally with *tangential Chow forms*.

MSC2010: primary 13P15; secondary 68W30.
Keywords: resultant, discriminant, Chow form.
Resultants version 1.2.1

In Section 1, from a more computational point of view, we give some background information on the general theory of resultants, discriminants, and Chow forms. In Section 2, we briefly illustrate how to use the package with the help of some examples; more detailed information and examples can be found in its documentation.

1. OVERVIEW OF CLASSICAL RESULTANTS. We present an overview of some classically well known facts on the theory of resultants for forms in several variables. For details and proofs, we refer mainly to [Gelfand et al. 1994; Cox et al. 2005]; other references are [Jouanolou 1991; 1997; van der Waerden 1950; Demazure 2012; Emiris and Mourrain 1999; Bajaj et al. 1988; Busé and Jouanolou 2014], and [Cox et al. 2007] for the case of two bivariate polynomials.

Resultants. Suppose we are given $n+1$ homogeneous polynomials $F_0, \dots, F_n$ in $n+1$ variables $x_0, \dots, x_n$ over the complex field $\mathbb{C}$. For $i = 0, \dots, n$, let d_i denote the total degree of F_i so that we can write $F_i = \sum_{|\alpha|=d_i} c_{i,\alpha} x^\alpha$, where x^α denotes $x_0^{\alpha_0} \cdots x_n^{\alpha_n}$. For each pair of indices i, α, we introduce a variable $u_{i,\alpha}$ and form the *universal ring of coefficients* $\mathbb{U}_{d_0,\dots,d_n} := \mathbb{Z}[u_{i,\alpha} : i = 0, \dots, n, |\alpha| = d_i]$. If $P \in \mathbb{U}_{d_0,\dots,d_n}$, we denote by $P(F_0, \dots, F_n)$ the element in $\mathbb{C}$ obtained by replacing each variable $u_{i,\alpha}$ with the corresponding coefficient $c_{i,\alpha}$.

Theorem 1.1 [Gelfand et al. 1994; Cox et al. 2005]. *If we fix positive degrees $d_0, \dots, d_n$, then there is a unique polynomial* $\mathrm{Res} = \mathrm{Res}_{d_0,\dots,d_n} \in \mathbb{U}_{d_0,\dots,d_n}$ *which has the following properties*:

(1) *If $F_0, \dots, F_n \in \mathbb{C}[x_0, \dots, x_n]$ are homogeneous of degrees $d_0, \dots, d_n$, then the equations*

$$F_0 = 0, \ \dots, \ F_n = 0$$

have a nontrivial solution over $\mathbb{C}$ (i.e., $\varnothing \neq V(F_0, \dots, F_n) \subset \mathbb{P}^n_{\mathbb{C}}$) if and only if $\mathrm{Res}(F_0, \dots, F_n) = 0$.

(2) Res *is irreducible, even when regarded as a polynomial over $\mathbb{C}$.*

(3) $\mathrm{Res}(x_0^{d_0}, \dots, x_n^{d_n}) = 1$.

Definition 1.2. We call $\mathrm{Res}(F_0, \dots, F_n)$ the *resultant* of $F_0, \dots, F_n$.

Remark 1.3. If A is any commutative ring, we define the resultant of $n+1$ homogeneous polynomials $F_0, \dots, F_n \in A[x_0, \dots, x_n]$ again as $\mathrm{Res}(F_0, \dots, F_n) \in A$, i.e., by specializing the coefficients of the integer polynomial Res. Thus, the formation of resultants commutes with specialization.

Example 1.4. The resultant is a direct generalization of the determinant. Indeed, if $d_0 = \cdots = d_n = 1$, then $\mathrm{Res}(F_0, \dots, F_n)$ equals the determinant of the $(n+1) \times (n+1)$ coefficient matrix.

Proposition 1.5 [Jouanolou 1991; Jouanolou 1997]. *The following hold*:

(1) (homogeneity) *For a fixed* j *between* 0 *and* n, Res *is homogeneous in the variables* $u_{j,\alpha}$, $|\alpha| = d_j$, *of degree* $d_0 \cdots d_{j-1} d_{j+1} \cdots d_n$; *hence its total degree is* $\sum_{j=0}^{n} d_0 \cdots d_{j-1} d_{j+1} \cdots d_n$.

(2) (symmetry) *If* σ *is a permutation of* $\{0, \ldots, n\}$, *then*

$$\operatorname{Res}(F_{\sigma(0)}, \ldots, F_{\sigma(n)}) = \operatorname{sign}(\sigma)^{d_0 \cdots d_n} \operatorname{Res}(F_0, \ldots, F_n).$$

(3) (multiplicativity) *If* $F_j = F'_j F''_j$, *then we have*

$$\operatorname{Res}(F_0, \ldots, F_j, \ldots, F_n) = \operatorname{Res}(F_0, \ldots, F'_j, \ldots, F_n) \operatorname{Res}(F_0, \ldots, F''_j, \ldots, F_n).$$

(4) (SL(n+1)-invariance) *For each* $(n+1) \times (n+1)$ *matrix* A *over* $\mathbb{C}$, *we have*

$$\operatorname{Res}(F_0(Ax), \ldots, F_n(Ax)) = \det(A)^{d_0 \cdots d_n} \operatorname{Res}(F_0(x), \ldots, F_n(x)),$$

where Ax *denotes the product of* A *with the column vector* $(x_0, \ldots, x_n)^t$.

(5) (elementary transformation) *If* H_i *is homogeneous of degree* $d_j - d_i$, *then*

$$\operatorname{Res}(F_0, \ldots, F_j + \sum_{i \neq j} H_i F_i, \ldots, F_n) = \operatorname{Res}(F_0, \ldots, F_j, \ldots, F_n).$$

Remark 1.6. On the product $\mathbb{A}^M \times \mathbb{P}^n = \operatorname{Spec}(\mathbb{C}[u_{i,\alpha}]) \times \operatorname{Proj}(\mathbb{C}[x_0, \ldots, x_n])$, where $M = \sum_{i=0}^{n} \binom{n+d_i}{n}$, we have an *incidence variety*

$$W := \left\{ ((c_{i,\alpha}), p) \in \mathbb{A}^M \times \mathbb{P}^n : p \in V\left(\sum_{|\alpha|=d_0} c_{0,\alpha} x^\alpha, \ldots, \sum_{|\alpha|=d_n} c_{n,\alpha} x^\alpha \right) \right\}.$$

The first projection $\pi_1 : W \to \mathbb{A}^M$ is birational onto its image, whereas all the fibers of the second projection $\pi_2 : W \to \mathbb{P}^n$ are linear subspaces of dimension $M - n - 1$. It follows that W is a smooth irreducible variety which is birational to $\overline{\pi_1(W)} = \pi_1(W) = V(\operatorname{Res}_{d_0,\ldots,d_n}) \subset \mathbb{A}^M$.

The following result is called the *Poisson formula* and allows one to compute resultants inductively.

Theorem 1.7 [Jouanolou 1991; Cox et al. 2005]. *Let*

$$f_i(x_0, \ldots, x_{n-1}) := F_i(x_0, \ldots, x_{n-1}, 1)$$

and $\overline{F_i}(x_0, \ldots, x_{n-1}) := F_i(x_0, \ldots, x_{n-1}, 0)$. *If* $\operatorname{Res}(\overline{F_0}, \ldots, \overline{F_{n-1}}) \neq 0$, *then the quotient ring* $A = \mathbb{C}[x_0, \ldots, x_{n-1}]/(f_0, \ldots, f_{n-1})$ *has dimension* $d_0 \cdots d_{n-1}$ *as a vector space over* $\mathbb{C}$, *and*

$$\operatorname{Res}(F_0, \ldots, F_n) = \operatorname{Res}(\overline{F_0}, \ldots, \overline{F_{n-1}})^{d_n} \det(m_{f_n} : A \to A), \tag{1-1}$$

where $m_{f_n} : A \to A$ *is the linear map given by multiplication by* f_n.

With the same hypotheses as Theorem 1.7, a monomial basis for A over $\mathbb{C}$ (useful in the implementation) can be constructed as explained in [Cox et al. 2005, Chapter 2, §2]. Note also that we have

$$\det(m_{f_n} : A \to A) = \prod_{p \in V} f_n(p)^{\mathrm{mult}_p(V)}, \tag{1-2}$$

where $V = V(f_0, \ldots, f_{n-1})$.

We now describe the most popular way to compute resultants, which is due to Macaulay [1903]. Let

$$\delta = \sum_{i=0}^{n} d_i - n \quad \text{and} \quad N = \binom{n+\delta}{n}.$$

We can divide the monomials x^α of total degree δ into the $n+1$ mutually disjoint sets

$$S_i := \{x^\alpha : |\alpha| = \delta,\ \min\{j : x_j^{d_j} | x^\alpha\} = i\}, \quad \text{for } i = 0, \ldots, n.$$

A monomial x^α of total degree δ is called *reduced* if $x_i^{d_i}$ divides x^α for exactly one i. Consider the following N homogeneous polynomials of degree δ:

$$x^\alpha / x_i^{d_i}\ F_i, \quad \text{for } i = 0, \ldots n \text{ and } x^\alpha \in S_i. \tag{1-3}$$

By regarding the monomials of total degree δ as unknowns, the polynomials in (1-3) form a system of N linear equations in N unknowns. Let

$$\mathbb{D} = \mathbb{D}(F_0, \ldots, F_n)$$

denote the coefficient matrix of this linear system, and let $\mathbb{D}'(F_0, \ldots, F_n)$ denote the submatrix of $\mathbb{D}$ obtained by deleting all rows and columns corresponding to reduced monomials. The following result is called the *Macaulay formula* and allows one to compute the resultant as a quotient of two determinants.

Theorem 1.8 [MacAulay 1903; Jouanolou 1997; Cox et al. 2005]. *The following formula holds*:

$$\det(\mathbb{D}(F_0, \ldots, F_n)) = \mathrm{Res}(F_0, \ldots, F_n)\ \det(\mathbb{D}'(F_0, \ldots, F_n)). \tag{1-4}$$

In several special cases, the resultant can be expressed as a single determinant (see [Gelfand et al. 1994, Chapter 13, Proposition 1.6]). We also mention that besides (1-4), there are other ways to represent resultants as quotients: these include *Bezoutians* [Elkadi and Mourrain 1998] and *Dixon matrices* [Kapur et al. 1994]; see also [Emiris and Mourrain 1999] and [Cox et al. 2005, p. 110]. However, all these matrices are usually of much larger size than those involved by the Poisson formula (1-1), as shown in the following simple example (see [Emiris and Mourrain 1999], for a comparison between Macaulay and other resultant matrices).

Example 1.9. Let $F_0 = x^3+y^2z,\ F_1 = xy+y^2+xz+yz,\ F_2 = y^4+z^4 \in \mathbb{C}[x, y, z]$. The Poisson formula expresses $\mathrm{Res}(F_0, F_1, F_2)$ as the following product of determinants:

$$\mathrm{Res}(F_0, F_1, F_2) = \left(1^2 \det\begin{pmatrix}1&0&0\\1&1&0\\0&1&1\end{pmatrix}\right)^4 \cdot \det\begin{pmatrix}1&0&0&0&0&0\\0&2&0&0&0&0\\0&0&2&0&0&0\\0&1&0&1&0&0\\1&0&0&0&2&0\\0&-1&0&1&0&2\end{pmatrix} = 16.$$

The Macaulay formula yields the same result as a quotient $\det(\mathbb{D})/\det(\mathbb{D}')$, where $\mathbb{D}$ and $\mathbb{D}'$ are square matrices of size 36×36 and 10×10, respectively.

Discriminants. Let $F = \sum_{|\alpha|=d} c_\alpha x^\alpha \in \mathbb{C}[x_0, \ldots, x_n]$ be a homogeneous polynomial of a certain degree d. As above, for each index α we introduce a variable u_α and form the universal ring of coefficients $\mathbb{U}_d := \mathbb{C}[u_\alpha : |\alpha| = d]$. Then one can show that, up to sign, there is a unique polynomial $\mathrm{Disc} = \mathrm{Disc}_d \in \mathbb{U}_d$ which has the following properties:

(1) If $F \in \mathbb{C}[x_0, \ldots, x_n]$ is homogeneous of degrees d, then the equations

$$\partial F/\partial x_0 = 0, \ldots, \partial F/\partial x_n = 0$$

have a nontrivial solution over $\mathbb{C}$ (i.e., the hypersurface defined by F is singular) if and only if $\mathrm{Disc}(F) = 0$;

(2) Disc is irreducible, even when regarded as a polynomial over $\mathbb{C}$.

Proposition 1.10 [Gelfand et al. 1994]. *Up to sign, we have the formula*

$$\mathrm{Disc}(F) = c_{d,n}\ \mathrm{Res}\left(\frac{\partial F}{\partial x_0}, \ldots, \frac{\partial F}{\partial x_n}\right),\ \textit{where } c_{d,n} = d^{\frac{(-1)^{n+1}-(d-1)^{n+1}}{d}}. \quad (1\text{-}5)$$

Definition 1.11. We call the polynomial defined by (1-5) the *discriminant* of F.

Proposition 1.12 [Gelfand et al. 1994]. *The following hold*:

(1) *The polynomial* Disc *is homogeneous of degree* $(n+1)(d-1)^n$.

(2) *For each* $(n+1)\times(n+1)$ *matrix* A *over* $\mathbb{C}$, *we have*

$$\mathrm{Disc}(F(Ax)) = \det(A)^{d(d-1)^n}\ \mathrm{Disc}(F(x)),$$

where Ax *denotes the product of* A *with the column vector* $(x_0, \ldots, x_n)^t$.

Geometrically, we have the following interpretation.

Proposition 1.13 [Gelfand et al. 1994]. *The discriminant hypersurface* $V(\mathrm{Disc}_d)$ *in the space of forms of degree* d *on* $\mathbb{P}^n$ *coincides with the dual variety of the* d*-th Veronese embedding of* $\mathbb{P}^n$.

Chow forms. Let $X \subset \mathbb{P}^n$ be an irreducible subvariety of dimension k and degree d. Consider the subvariety $Z(X)$ in the Grassmannian $\mathbb{G}(n-k-1, \mathbb{P}^n)$ of all $(n-k-1)$-dimensional projective subspaces of $\mathbb{P}^n$ that intersect X. It turns out that $Z(X)$ is an irreducible hypersurface of degree d; thus $Z(X)$ is defined by the vanishing of some element R_X, unique up to a constant factor, in the homogeneous component of degree d of the coordinate ring of the Grassmannian $\mathbb{G}(n-k-1, \mathbb{P}^n)$ in the Plücker embedding. This element is called the *Chow form* of X. It is notable that X can be recovered from its Chow form. See [Gelfand et al. 1994, Chapter 3, §2] for details.

Consider the product $\mathbb{P}^k \times X$ as a subvariety of $\mathbb{P}^{(k+1)(n+1)-1}$ via the Segre embedding. Identify $\mathbb{P}^{(k+1)(n+1)-1}$ with the projectivization $\mathbb{P}(\mathrm{Mat}(k+1, n+1))$ of the space of $(k+1) \times (n+1)$ matrices and consider the natural projection $\rho : \mathbb{P}(\mathrm{Mat}(k+1, n+1)) \dashrightarrow \mathbb{G}(k, n) \simeq \mathbb{G}(n-k-1, n)$. The following result is called the *Cayley trick*.

Theorem 1.14 [Gelfand et al. 1994; Weyman and Zelevinsky 1994]. *The dual variety of $\mathbb{P}^k \times X$ coincides with the closure $\overline{\rho^{-1}(Z(X))}$, where*

$$Z(X) \subset \mathbb{G}(n-k-1, n)$$

is the hypersurface defined by the Chow form of X.

The defining polynomial of the hypersurface $\overline{\rho^{-1}(Z(X))} \subset \mathbb{P}(\mathrm{Mat}(k+1, n+1))$ is called *X-resultant*; it provides another way of writing the Chow form of X.

Now, let $F_0, \ldots, F_n$ be $n+1$ generic homogeneous polynomials on $\mathbb{P}^n$ of the same degree $d > 0$, and let $\mathbb{M} = \mathbb{M}(F_0, \ldots, F_n)$ be the $(n+1) \times N$ matrix of the coefficients of these polynomials, $N = \binom{n+d}{n}$. We consider the projection $\rho_{n,d} : \mathbb{P}(\mathrm{Mat}(n+1, N)) \dashrightarrow \mathbb{G}(n, N-1) \simeq \mathbb{G}(N-n-2, N-1)$ defined by the maximal minors of $\mathbb{M}$.

Proposition 1.15 [Gelfand et al. 1994; Cox et al. 2005]. *The hypersurface of degree $(n+1)d^n$ in $\mathbb{P}(\mathrm{Mat}(n+1, N))$ defined by the resultant* $\mathrm{Res}(F_0, \ldots, F_n)$ *coincides with the closure $\overline{\rho_{n,d}^{-1}(V(R_{n,d}))}$, where $R_{n,d}$ denotes the Chow form of the d-th Veronese embedding of $\mathbb{P}^n$. In particular,* $\mathrm{Res}(F_0, \ldots, F_n)$ *is a polynomial in the maximal minors of $\mathbb{M}$.*

2. IMPLEMENTATION. In this section, we illustrate briefly some of the methods available in the package *Resultants*, included with [Macaulay2]. We refer to the package documentation (which can be viewed with `viewHelp Resultants`) for more details and examples.

One of the main methods is `resultant`, which accepts as input a list of $n+1$ homogeneous polynomials in $n+1$ variables with coefficients in some commutative ring A and returns an element of A, the resultant of the polynomials. There are

no limitations on the ring A because of Remark 1.3. The algorithms implemented are the Poisson formula (Theorem 1.7) and the Macaulay formula (Theorem 1.8). The former is used by default since it is typically faster, while for the latter one has to set the Algorithm option: `resultant(...,Algorithm=>"Macaulay")`. The method can also be configured to involve interpolation of multivariate polynomials (see [Manocha and Canny 1993]), i.e., it can reconstruct the polynomial resultant from its values at a sufficiently large number of points, which in turn are evaluated using the same formulas. The main derived method is `discriminant`, which applies the formula (1-5) to compute discriminants of homogeneous polynomials.

Example 2.1. In the following code, we take two forms F, G of degree 6 on $\mathbb{P}^3$. We first verify that $\mathrm{Disc}(F) = 0$ and $\mathrm{Disc}(G) \neq 0$ and then we compute the intersection of the pencil generated by F and G with the discriminant hypersurface in the space of forms of degree 6 on $\mathbb{P}^3$, which is a hypersurface of degree 500 in $\mathbb{P}^{83}$. (The algorithm behind these calculations is the Poisson formula; this is one of the cases where the Macaulay formula is much slower).

```
Macaulay2, version 1.10
with packages: ConwayPolynomials, Elimination, IntegralClosure, InverseSystems,
               LLLBases, PrimaryDecomposition, ReesAlgebra, TangentCone
i1 : loadPackage "Resultants";
i2 : ZZ[w,x,y,z]; (F,G) = (w^6+x^6+y^6+w*x*y^4,w^6+x^6+y^6+z^6)
       6    6        4    6   6    6    6    6
o3 = (w  + x  + w*x*y  + y , w  + x  + y  + z )
o3 : Sequence
i4 : time discriminant F
     -- used 0.0179806 seconds
o4 = 0
i5 : time discriminant G
     -- used 0.0310744 seconds
o5 = 14057081148316919947063801793278835854428218771739784465632482676955216027847633256406502145120855236676811697488882435760217714078399664105019672381338748228576388016904232984135762316136175977862452217324448345919411204360245828922074151228959163773714466361681597648097658753070739833449997864683601657856
i6 : R := ZZ[t,u][w,x,y,z]; pencil = t*sub(F,R) + u*sub(G,R)
             6          6          4          6      6
o7 = (t + u)w  + (t + u)x  + t*w*x*y  + (t + u)y  + u*z
o7 : ZZ[t, u][w, x, y, z]
i8 : time D = discriminant pencil
     -- used 7.05101 seconds
                                    375 125                                   374 126 ...
     11918167904272470982401...000t   u    + 44811489377450403137211...000t   u    ...
o8 : ZZ[t, u]
i9 : factor D
        125       195    3      2         2      3 30    3      2         2      3 30
o9 = (u)   (t + u)   (25t  + 81t u + 81t*u  + 27u )  (29t  + 81t u + 81t*u  + 27u )  (
1845309860334485435604513007620143382090608492211798740863140403531458335493678485869019666668055428407222803144055042891867966935429959336227999512218285981355846846846364626801397625813957058058834010980828766582924640256)
o9 : Expression of class Product
```

In particular, we deduce that the pencil $\langle F, G\rangle$ intersects the discriminant hypersurface in F with multiplicity 125, in $F - G$ with multiplicity 195, and in another six distinct points with multiplicity 30.

The package also provides methods for working with Chow forms and more generally tangential Chow forms of projective varieties (see [Gelfand et al. 1994, p. 104] and [Green and Morrison 1986]). In the following example, we apply some of these methods.

Example 2.2. Take $C \subset \mathbb{P}^3$ to be the twisted cubic curve.

```
i10 : C = kernel veronese(1,3)
              2                       2
o10 = ideal (x  - x x , x x  - x x , x  - x x )
              2    1 3   1 2    0 3   1    0 2

o10 : Ideal of QQ[x , x , x , x ]
                  0   1   2   3
```

The Chow form of C in $\mathbb{G}(1, 3)$ can be obtained as follows:

```
i11 : w = chowForm C
       3                          2      2
o11 = x    - x   x   x    + x   x    + x   x    - 2x   x   x    - x   x   x
       1,2    0,2 1,2 1,3    0,1 1,3    0,2 2,3     0,1 1,2 2,3    0,1 0,3 2,3

      QQ[x   , x   , x   , x   , x   , x   ]
         0,1   0,2   1,2   0,3   1,3   2,3
o11 : ---------------------------------------
          x   x    - x   x    + x   x
           1,2 0,3    0,2 1,3    0,1 2,3
```

We can recover C from its Chow form by taking the so-called Chow equations; see [Gelfand et al. 1994, p. 102; Catanese 1992].

```
i12 : C == saturate chowEquations w
o12 = true
```

The X-resultant of C can be obtained applying first the duality isomorphism $\mathbb{G}(1, \mathbb{P}^3) = \mathbb{G}(1, \mathbb{P}^{3*})$ and then passing from the Plücker to the Stiefel coordinates.

```
i13 : w' = dualize w
       3                          2      2
o13 = x    - x   x   x    + x   x    + x   x    - x   x   x    - 2x   x   x
       0,3    0,2 0,3 1,3    0,1 1,3    0,2 2,3    0,1 1,2 2,3     0,1 0,3 2,3

      QQ[x   , x   , x   , x   , x   , x   ]
         0,1   0,2   1,2   0,3   1,3   2,3
o13 : ---------------------------------------
          x   x    - x   x    + x   x
           1,2 0,3    0,2 1,3    0,1 2,3

i14 : fromPluckerToStiefel w'
```

```
          3   3          2   2              2       2          2   3      ...
o14 = - x   x    + x   x   x   x    - x   x   x   x    + x   x   x    - ...
        0,3 1,0    0,2 0,3 1,0 1,1    0,1 0,3 1,0 1,1    0,0 0,3 1,1    ...

o14 : QQ[x   , x   , x   , x   , x   , x   , x   , x   ]
         0,0   0,1   0,2   0,3   1,0   1,1   1,2   1,3
```

The method `cayleyTrick` returns a pair consisting of the defining ideal of $\mathbb{P}^1 \times C \subset \mathbb{P}^7 \simeq \mathbb{P}(\mathrm{Mat}(2,4))$ and the X-resultant of C, considered as a hypersurface $Z \subset \mathbb{P}(\mathrm{Mat}(2,4))$. Theorem 1.14 ensures that Z is the dual variety of $\mathbb{P}^1 \times C$. We can check this using the method `dualVariety`.

```
i15 : (P1xC,Z) = cayleyTrick C;
i16 : dualVariety(P1xC) == Z
o16 = true
```

Some overlapping packages. There are two further packages related to resultant computations, which are included in Macaulay2: [Elimination] by M. E. Stillman, and [EliminationMatrices] by N. Botbol, L. Busé and M. Dubinsky. The former contains functions to compute Sylvester resultants. The latter can compute different resultant matrices; in particular, it contains an implementation of the Macaulay formula.

A further package for working with Chow forms is *Coisotropy*, by K. Kohn (see [Kohn 2016]), which, in particular, contains a useful function to compute the degrees of all tangential Chow forms of a given projective variety.

For all these overlapping functions, it does not seem easy to rank implementations in terms of efficiency because this generally depends on the problem. They also differ in how they handle input and output. For instance, the discriminant of a binary form computed using the package *Elimination* lies again in the same ring, rather than in the ring of coefficients, and the Chow form of a projective variety computed using *Coisotropy* lies in a polynomial ring, rather than in a quotient ring.

SUPPLEMENT. The online supplement contains version 1.2.1 of *Resultants*.

REFERENCES.

[Bajaj et al. 1988] C. Bajaj, T. Garrity, and J. Warren, "On the applications of mulit-equational resultants", tech report 88-826, Purdue University, 1988.

[Busé and Jouanolou 2014] L. Busé and J.-P. Jouanolou, "On the discriminant scheme of homogeneous polynomials", *Math. Comput. Sci.* **8**:2 (2014), 175–234. MR Zbl

[Catanese 1992] F. Catanese, "Chow varieties, Hilbert schemes and moduli spaces of surfaces of general type", *J. Algebraic Geom.* **1**:4 (1992), 561–595. MR Zbl

[Cox et al. 2005] D. A. Cox, J. Little, and D. O'Shea, *Using algebraic geometry*, 2nd ed., Graduate Texts in Mathematics **185**, Springer, 2005. MR Zbl

[Cox et al. 2007] D. Cox, J. Little, and D. O'Shea, *Ideals, varieties, and algorithms: An introduction to computational algebraic geometry and commutative algebra*, 3rd ed., Springer, New York, 2007. MR Zbl

[Demazure 2012] M. Demazure, "Résultant, discriminant", *Enseign. Math.* (2) **58**:3-4 (2012), 333–373. MR Zbl

[Elimination] M. E. Stillman, "Elimination", Macaulay2 package, 2005, available at http://github.com/Macaulay2/M2/blob/master/M2/Macaulay2/packages/Elimination.m2.

[EliminationMatrices] N. Botbol, L. Busé, and M. Dubinksy, "EliminationMatrices", Macaulay2 package, 2012, available at http://github.com/Macaulay2/M2/blob/master/M2/Macaulay2/packages/EliminationMatrices.m2.

[Elkadi and Mourrain 1998] M. Elkadi and B. Mourrain, "Some applications of Bezoutians in effective algebraic geometry", INRIA, 1998, available at https://hal.inria.fr/inria-00073109.

[Emiris and Mourrain 1999] I. Z. Emiris and B. Mourrain, "Matrices in elimination theory", *J. Symbolic Comput.* **28**:1-2 (1999), 3–44. MR Zbl

[Gelfand et al. 1994] I. M. Gelfand, M. M. Kapranov, and A. V. Zelevinsky, *Discriminants, resultants, and multidimensional determinants*, Birkhäuser, Boston, 1994. MR Zbl

[Green and Morrison 1986] M. L. Green and I. Morrison, "The equations defining Chow varieties", *Duke Math. J.* **53**:3 (1986), 733–747. MR Zbl

[Jouanolou 1991] J.-P. Jouanolou, "Le formalisme du résultant", *Adv. Math.* **90**:2 (1991), 117–263. MR Zbl

[Jouanolou 1997] J. P. Jouanolou, "Formes d'inertie et résultant: un formulaire", *Adv. Math.* **126**:2 (1997), 119–250. MR Zbl

[Kapur et al. 1994] D. Kapur, T. Saxena, and L. Yang, "Algebraic and geometric reasoning using Dixon resultants", pp. 99–107 in *Proceedings of the International Symposium on Symbolic and Algebraic Computation*, ACM, New York, NY, USA, 1994. Zbl

[Kohn 2016] K. Kohn, "Coisotropic hypersurfaces in grassmannians", 2016. arXiv

[MacAulay 1903] F. S. MacAulay, "On some formulae in elimination", *Proc. Lond. Math. Soc.* **35** (1903), 3–27. MR Zbl

[Macaulay2] D. R. Grayson and M. E. Stillman, "Macaulay2, a software system for research in algebraic geometry", software, version 1.10, available at http://faculty.math.illinois.edu/Macaulay2.

[Manocha and Canny 1993] D. Manocha and J. F. Canny, "Multipolynomial resultant algorithms", *J. Symbolic Comput.* **15**:2 (1993), 99–122. MR Zbl

[van der Waerden 1950] B. L. van der Waerden, *Modern algebra*, vol. II, Ungar, New York, 1950. Reprinted Springer, 1991, as *Algebra*, vol. II. MR

[Weyman and Zelevinsky 1994] J. Weyman and A. Zelevinsky, "Multiplicative properties of projectively dual varieties", *Manuscripta Math.* **82**:2 (1994), 139–148. MR Zbl

RECEIVED: 3 May 2017 REVISED: 26 Jan 2018 ACCEPTED: 18 May 2018

GIOVANNI STAGLIANÒ:

giovannistagliano@gmail.com

Dipartimento di Ingegneria Industriale e Scienze Matematiche, Università Politecnica delle Marche, Ancona, Italy

JSAG **8** (2018), 31–48
dx.doi.org/10.2140/jsag.2018.8.31

The Journal of Software for
Algebra and Geometry

The SpaceCurves package in Macaulay2

MENGYUAN ZHANG

ABSTRACT: This note introduces the Macaulay2 package `SpaceCurves.m2` with illustration. The 1.0 version of the package, provided in the accompanying online supplement, is devoted to the generation of three types of curves in $\mathbb{P}^3$: smooth curves, ACM curves and curves that are minimal in the even liaison class.

1. INTRODUCTION. The `SpaceCurves` project was initiated by R. Hartshorne, F. Schreyer and M. Stillman during the 2017 Macaulay2 workshop at UC Berkeley. Since the workshop, Zhang has improved old code and developed the package into the present version. The goal of the `SpaceCurves` package is to generate three types of curves in $\mathbb{P}^3$; smooth curves, ACM curves, and minimal curves in a given even liaison class.

In Section 2 we illustrate how to produce smooth curves exhausting all possibilities of (degree, genus) pairs. The philosophy is the following: first we construct three types of surfaces; the smooth quadric surface, smooth cubic surfaces and rational quartic surfaces with a double line. Next, we construct divisors on these surfaces. Finally, we generate a random curve in a given divisor class.

In Section 3 we illustrate the stratification of the Hilbert scheme of ACM curves in $\mathbb{P}^3$ using Betti tables. We illustrate how to produce ACM curves exhausting all possibilities of Betti tables. First, we list all the possible Hilbert functions of ACM curves of a given degree, then we produce all Betti tables of ACM curves with a given Hilbert function. Finally, we generate a matrix of random forms with degrees specified by the Hilbert–Burch degree matrix and take the ideal of maximal minors.

In Section 4 we explain the construction of a curve that is minimal in its even liaison class from a given finite length module. The even liaison class can be specified by either the ideal of a curve in the even liaison class, or by a finite length module called the Hartshorne–Rao module. The implementation of the minimal curve algorithm follows the paper by Guarrera et al. [1997].

To make our computations exact, and to avoid coefficient explosion, we work over large finite fields $\mathbb{Z}/p\mathbb{Z}$ in Macaulay2. Over a small prime field, such as $\mathbb{Z}/2\mathbb{Z}$

MSC2010: 14-04, 14H50, 14HXX.
Keywords: space curves, minimal curves, Rao module, liaison theory, ACM curves, smooth curves.
SpaceCurves version 1.0

or $\mathbb{Z}/3\mathbb{Z}$, many of the curves produced will be singular. In these cases we refer the readers to the package `RandomCurvesOverVerySmallFiniteFields.m2` by C. Bopp and F. Schreyer.

2. SMOOTH CURVES IN PROJECTIVE THREE-SPACE. For which pairs of integers (d, g) does there exist a connected smooth curve in $\mathbb{P}^3$ of degree d and genus g? G. Halphen gave partial answers in his prize winning treatise in 1882, and the complete solution to this question was given by Gruson and Peskine almost a century later in characteristic 0 and extended to characteristic p by Hartshorne.

(1) There are smooth plane curves of genus $g = \frac{1}{2}(d-1)(d-2)$ for any $d \geq 1$.

(2) (Castelnuovo) If a smooth curve does not lie on any plane, we must have
$$g \leq \left\lfloor \tfrac{1}{4}d^2 - d + 1 \right\rfloor.$$
Any curve obtaining this bound lies on a quadric surface.

(3) For each $a, b > 0$, there are smooth curves on the smooth quadric surface with degree $d = a + b$ and genus $g = (a-1)(b-1)$.

(4) On the quadric cone, if $d = 2a$ is even, there are smooth complete intersections of the quadric cone with another surface of degree a. If $d = 2a + 1$ is odd, then any degree d curve on the quadric cone has genus $g = a^2 - a$.

(5) [Halphen 1882] If a curve does not lie on any plane or quadric surface, then
$$g \leq \tfrac{1}{6}d(d-3) + 1.$$

(6) [Gruson and Peskine 1982, Corollary 2.3] For $d \geq 1$ and
$$\tfrac{1}{\sqrt{3}}d^{3/2} - d + 1 < g \leq \tfrac{1}{6}d(d-3) + 1,$$
there is a smooth curve with degree d and genus g on a smooth cubic surface.

(7) [Gruson and Peskine 1982, Theorem 1.1; Hartshorne 1982, Theorem 0.2] For $d \geq 1$ and
$$0 \leq g \leq \tfrac{1}{8}(d-1)^2,$$
there is a smooth curve with degree d and genus g on a smooth quartic surface with a double line.

The `SpaceCurves` package generates curves on the surfaces mentioned above. Let us start from curves on the smooth quadric surface.

2.1. ***Curves on a smooth quadric surface.*** To create a smooth quadric surface Q, we use the method function `quadricSurface(Ring)` where the input ring is taken as the ambient coordinate ring of $\mathbb{P}^3$. The output is a type of hashtable called `QuadricSurface`. It contains the following information:

```
i1 : needsPackage "SpaceCurves";
i2 : R = ZZ/101[x_0..x_3];
i3 : Q = quadricSurface(R)
o3 = ideal(- x x  + x x )
             1 2    0 3
o3 : QuadricSurface
i4 : peek Q
o4 = QuadricSurface{CanonicalClass => {-2, -2}    }
                    HyperplaneClass => {1, 1}
                    Ideal => ideal(- x x  + x x )
                                     1 2    0 3
                    IntersectionPairing => | 0 1 |
                                           | 1 0 |
```

Since $Q \cong \mathbb{P}^1 \times \mathbb{P}^1$, a basis of $\mathrm{Pic}(Q)$ consists of $\pi_1^*(\mathscr{O}_{\mathbb{P}^1}(1))$ and $\pi_2^*(\mathscr{O}_{\mathbb{P}^1}(1))$, where π_1 and π_2 are the two canonical projections to $\mathbb{P}^1$. In this basis, the intersection pairing matrix is $\left[\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right]$ and the canonical class has coordinates $\{-2, -2\}$. The divisor class $\{1, 1\}$ in the given basis defines the embedding of Q in $\mathbb{P}^3$.

The method function `divisor(List,QuadricSurface)` produces divisors on the quadric surface Q. The output is a type of hashtable called `Divisor`, which carries a list that encodes the coordinates of the divisor as well as the surface it is on. The intersection number is computed from the coordinates of D and E using the intersection matrix of Q.

```
i5 : D = divisor({2,3},Q)
o5 = {2, 3}
o5 : Divisor
i6 : peek D
o6 = Divisor{Coordinate => {2, 3}         }
             Surface => ideal(- y*z + x*w)
i7 : E = divisor({1,2},Q);
i8 : D*E
o8 = 7
```

Since we can compute the intersection number of any two divisors on Q, we can compute the degree and arithmetic genus of a divisor using Bezout's theorem and the adjunction formula:

$$\deg D = D.H,$$
$$2p_a(D) - 2 = D.(D + K),$$

where H and K denote the hyperplane class and the canonical class, respectively. The advantage of computing the degree and genus of a divisor abstractly is that the coordinates of the divisor can be in any ring that supports rational arithmetic. In particular, we can verify the formula for the degree and genus of a divisor of type (a, b) on Q.

```
i9 : S = QQ[a,b];
i10 : F = divisor({a,b},Q);
i11 : degree F
o11 = a + b
o11 : S
i12 : genus F
o12 = a*b - a - b + 1
o12 : S
```

We use the method function `curve(Divisor)` to produce a random curve in a given divisor class. The output type is called `Curve`, which is a hashtable that encodes the ideal of the curve as well as the `Divisor` it comes from. To extract the ideal, one could use the method `ideal(Curve)` or equivalently the key `Ideal` of the hashtable `Curve`.

```
i13 : C = curve D;
i14 : I = ideal C;
i15 : (degree I, degree D, genus I, genus D)
o15 = (5, 5, 2, 2)
o15 : Sequence
```

Here is how the method `curve(Divisor)` works for divisors on a smooth quadric surface. If D has coordinates $\{a, b\}$, we generate a random form f of bidegree (a, b) in the Cox ring

$$\mathrm{Cox} = k[s, t] \otimes_k k[u, v]$$

and create the Segre map $\psi : R = k[x, y, z, w] \to \mathrm{Cox}/(f)$ where

$$x \mapsto s \otimes u, \quad y \mapsto s \otimes v, \quad z \mapsto t \otimes u, \quad w \mapsto t \otimes v.$$

The kernel of the map ψ will be the ideal of a curve in the divisor class (a, b) on the quadric surface. Since we are using random bihomogeneous forms, the output is typically smooth.

2.2. ***Curves on a smooth cubic surface.*** Although it is easy to produce a smooth cubic surface, it is not straightforward to generate curves of a given degree and genus on the given cubic surface. Instead, we realize the cubic surface as a blowup $\pi : X \to \mathbb{P}^2$ at six general points, anticanonically embedded into $\mathbb{P}^3$. Since the automorphisms of $\mathbb{P}^2$ act transitively on four distinct points, we fix four points,

$$P_1 = [1 : 0 : 0], \quad P_2 = [0 : 1 : 0], \quad P_3 = [0 : 0 : 1], \quad P_4 = [1 : 1 : 1],$$

and choose points P_5 and P_6 randomly. Then $\mathrm{Pic}(X) \cong \mathbb{Z}^7$ with a basis given by $L, -E_1, \ldots, -E_6$, where $L = \pi^*(\mathscr{O}_{\mathbb{P}^2}(1))$ and E_i is the exceptional divisor of the

point P_i. The intersection matrix of X in this basis is given by

$$\begin{bmatrix} 1 & 0 \\ 0 & -\mathrm{Id} \end{bmatrix},$$

where Id is the 6×6 identity matrix. The complete linear system $|3L-E_1-\cdots-E_6|$ is very ample and embeds X into $\mathbb{P}^3$ as a smooth cubic surface. The canonical class of X is $-3L+E_1+\cdots+E_6$. See [Hartshorne 1977, V.4] for a treatment of these well known facts.

We can use the method function `cubicSurface(Ring)` to generate a smooth cubic surface. The output is a type of hashtable called `CubicSurface`. The key `BlowUpPoints` stores the list of ideals of the six points. The key `MapToP3` stores the rational map from $\mathbb{P}^2$ to $\mathbb{P}^3$ which comes from the restriction of the embedding $X \hookrightarrow \mathbb{P}^3$.

```
i5 : X = cubicSurface(R)
o5 = ideal(...)
o5 : CubicSurface
i6 : peek X
o6 = CubicSurface{BlowUpPoints => {...}                          }
                  CanonicalClass => {-3, -1, -1, -1, -1, -1, -1}
                  HyperplaneClass => {3, 1, 1, 1, 1, 1, 1}
                  IntersectionPairing => | 1 0  0  0  0  0  0  |
                                         | 0 -1 0  0  0  0  0  |
                                         | 0 0  -1 0  0  0  0  |
                                         | 0 0  0  -1 0  0  0  |
                                         | 0 0  0  0  -1 0  0  |
                                         | 0 0  0  0  0  -1 0  |
                                         | 0 0  0  0  0  0  -1 |
                  Ideal => ideal(...)
                  MapToP3 => ...
```

The equation of X in $\mathbb{P}^3$ is computed in the following way: let I be the ideal of the union of the six points $P_1,\ldots,P_6$. Since P_5 and P_6 are chosen randomly, we can presume that no three of P_i are collinear and that the points P_i do not all lie on any quadric. The Hilbert–Burch theorem thus determines the shape of the free resolution of I as

$$0 \to R(-4)^3 \xrightarrow{\phi} R(-3)^4 \to I \to 0,$$

where $R = k[y_0, y_1, y_2]$ is the coordinate ring of $\mathbb{P}^2$. Now ϕ is a 4×3 matrix of linear forms in three variables, which can be thought of as a $4\times 3\times 3$ tensor. There is a canonical way to consider ϕ as a $3\times 3\times 4$ tensor, and think of it as a 3×3 matrix of linear forms in four variables denoted by M'. The determinant of M' gives the defining equation of X in $\mathbb{P}^3$.

We can create divisors on X using `divisor(List,CubicSurface)`. Let us compute the degree and genus of a divisor on the smooth cubic surface.

```
i17 : S = QQ[a,b_1..b_6];
i18 : D = divisor(gens S,X);
i19 : degree D
o19 = 3a - b  - b  - b  - b  - b  - b
          1    2    3    4    5    6
i20 : genus D
o20 = ...
```

2.3. ***Curves on a rational quartic surface.*** Let us recall the construction of the quartic surface with a double line singularity from [Gruson and Peskine 1982]. Consider the blowup of $\mathbb{P}^2$ at nine general points $P_1, \ldots, P_9$ given by $\pi : Y \to \mathbb{P}^2$. Let L denote the class $\pi^*(\mathscr{O}^1_{\mathbb{P}}(1))$ on Y, and let E_i denote the class of the exceptional divisor corresponding to the point P_i. Then $\mathrm{Pic}(Y)$ is isomorphic to the free abelian group generated by $L, -E_1, \ldots, -E_9$. There is a unique smooth cubic curve Γ_0 on $\mathbb{P}^2$ passing through the nine points, and let Γ denote its proper transform on Y. We have $\Gamma = 3L - E_1 - \cdots - E_9$, and $-\Gamma$ is the canonical class of Y. If we set $C = L - E_1$, then the complete linear system $|C + \Gamma|$ is basepoint-free and maps Y into $\mathbb{P}^3$ as a quartic hypersurface. One can verify that the linear system $|C + \Gamma|$ separates points and tangent vectors for points in $Y - \Gamma$, and restricts to a degree 2 linear system on the elliptic curve Γ. As a result, $Y - \Gamma$ is mapped isomorphically onto its image, and Γ is mapped 2-1 to a line. The image of Y is a rational quartic surface with a double line singularity, where the double line is the scheme-theoretical image of the elliptic curve Γ. For proofs of these statements, see [Gruson and Peskine 1982, §1].

We use the method function `quarticSurfaceRational(Ring)` to create such a surface. We can verify that the singular locus of the image of Y is indeed the image of the elliptic Γ in Macaulay2.

```
i21 : Y = quarticSurfaceRational(R)
o21 = ideal(...)
o21 : QuarticSurfaceRational
i22 : peek Y
o22 = QuarticSurfaceRational{BlowUpPoints => {...
        CanonicalClass => {-3, -1, -1, -1, -1, -1, -1, -1, -1, -1}
        HyperplaneClass => {4, 2, 1, 1, 1, 1, 1, 1, 1, 1}
        IntersectionPairing => | 1 0  0  0  0  0  0  0  0  0  |
                               | 0 -1 0  0  0  0  0  0  0  0  |
                               | 0 0  -1 0  0  0  0  0  0  0  |
                               | 0 0  0  -1 0  0  0  0  0  0  |
                               | 0 0  0  0  -1 0  0  0  0  0  |
                               | 0 0  0  0  0  -1 0  0  0  0  |
                               | 0 0  0  0  0  0  -1 0  0  0  |
                               | 0 0  0  0  0  0  0  -1 0  0  |
                               | 0 0  0  0  0  0  0  0  -1 0  |
                               | 0 0  0  0  0  0  0  0  0  -1 |
```

```
        MapToP3 => ...
        Ideal => ...
i26 : pts = intersect(Y.BlowUpPoints);
i31 : S = ring pts;
i27 : hilbertFunction(3,module pts)
o27 = 1
i28 : G = gens trim pts;
i32 : gamma0 = ideal (G*random(source G,S^{-3}));
      -- This produces the unique plane cubic passing the 9 points.
o32 : Ideal of S
i36 : phi = map(S/gamma0,S);
i37 : psi = Y.MapToP3;
i39 : L = kernel (phi*psi)
o39 = ideal (y + 9918z - 2540w, x + 2540z - 11757w)
o39 : Ideal of R
      -- This is the image of Gamma
i44 : radical ideal jacobian Y.Ideal
o44 = ideal (y + 9918z - 2540w, x + 2540z - 11757w)
o44 : Ideal of R
      -- This is the singular locus of the image of Y
i45 : radical ideal jacobian Y.Ideal == L
o45 = true
```

The method function `divisor(List,QuarticSurfaceRational)` creates a divisor on the quartic surface with a double line. We explain how the method function `curve(Divisor)` turns `Divisors` into `Curves` on the `CubicSurface` and the `QuarticSurfaceRational`. Given a divisor $D = (a, b_1, \ldots, b_n)$ where $n = 6$ or 9, we generate a random plane curve C_0 of degree a with multiplicity b_i at the point P_i and compute the equations of its image under the rational map

$$\mathbb{P}^2 \dashrightarrow \mathbb{P}^3.$$

Note that the produced curves have no components supported on the exceptional curves.

2.4. ***Generating smooth curves.*** We say a divisor class is smooth if its general members have smooth images in $\mathbb{P}^3$. The method function `smoothDivisors` generates all smooth divisors of a given degree on a given surface. So far the valid input surfaces are `QuadricSurface`, `CubicSurface` and `QuarticSurfaceRational`. On the smooth quadric surface as well as the smooth cubic surface, there are exact numerical criteria to decide whether D is smooth [Hartshorne 1977, Example V.4.8]. On the rational quartic surface however, we do not know of such a criterion, and thus our list is not exhaustive. One sufficient condition for D to be smooth is that D is basepoint-free and its restriction to Γ does not contain any pair of involution points as basepoints.

The following code generates smooth divisors of degree 4 on all the supported surfaces. Note that the output is a list of `Divisors`.

```
i2 : R = ZZ/32003[x_0..x_3];
i3 : Q = quadricSurface(R); X = cubicSurface(R);
i5 : Y = quarticSurfaceRational(R);
i6 : L = smoothDivisors(4,Q) | smoothDivisors(4,X) | smoothDivisors(4,Y);
i7 : netList L
     +------------------------------+
o7 = |{1, 3}                       |
     +------------------------------+
     |{2, 2}                       |
     +------------------------------+
     |{2, 1, 1, 0, 0, 0, 0}        |
     +------------------------------+
     |{3, 1, 1, 1, 1, 1, 0}        |
     +------------------------------+
     |{2, 0, 1, 1, 1, 1, 0, 0, 0, 0}|
     +------------------------------+
     |{3, 1, 1, 1, 1, 1, 1, 1, 0, 0}|
     +------------------------------+
```

Now we take the list of `Divisors` and turn them into `Curves`, and verify that the curves we get are indeed smooth and have the correct degrees and genera.

```
i8 : LC = apply(L,D->curve D);
i9 : netList apply(LC,C-> (degree C,genus C, isSmooth C))
     +------------+
o9 = |(4, 0, true)|
     +------------+
     |(4, 1, true)|
     +------------+
     |(4, 0, true)|
     +------------+
     |(4, 1, true)|
     +------------+
     |(4, 0, true)|
     +------------+
     |(4, 1, true)|
     +------------+
```

The method function `curve(ZZ,ZZ)` takes a pair of integers (d, g), generates divisors of degree d on all the surfaces, and selects one (if any) of genus g and returns it as a `Curve`.

```
i11 : C = curve(8,5);
i12 : degree C, genus C
o12 = (8, 5)
o12 : Sequence
```

The following code generated 608 curves up to degree 15 in around three minutes on a desktop with quad cores Intel i5-7400 CPU at 3.00GHz.

```
i2 : R = ZZ/101[x_0..x_3];
i3 : X = quadricSurface(R);
i4 : Y = cubicSurface(R);
i5 : Z = quarticSurfaceRational(R);
i6 : dmax = 15;
i7 : time LD = flatten apply(splice{1..dmax},d->
         smoothDivisors(d,X) | smoothDivisors(d,Y) | smoothDivisors(d,Z));
     -- used 2.3459 seconds
i8 : time LC = apply(LD,D -> curve D);
     -- used 185.549 seconds
i9 : #LC
o9 = 608
```

We use the method function `dgTable` to tally the number of curves by the degree (horizontal axis) and genus (vertical axis).

```
i10 : dgTable LC
o10 = 42 |  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  1
      41 |  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
      40 |  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  1
      39 |  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
      38 |  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
      37 |  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
      36 |  .  .  .  .  .  .  .  .  .  .  .  .  .  .  1  1
      35 |  .  .  .  .  .  .  .  .  .  .  .  .  .  .  1  .
      34 |  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
      33 |  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
      32 |  .  .  .  .  .  .  .  .  .  .  .  .  .  .  1  .
      31 |  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  1
      30 |  .  .  .  .  .  .  .  .  .  .  .  .  .  1  .  2
      29 |  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  1
      28 |  .  .  .  .  .  .  .  .  .  .  .  .  .  1  .  1
      27 |  .  .  .  .  .  .  .  .  .  .  .  .  .  .  1  3
      26 |  .  .  .  .  .  .  .  .  .  .  .  .  .  .  1  1
      25 |  .  .  .  .  .  .  .  .  .  .  .  .  1  .  1  3
      24 |  .  .  .  .  .  .  .  .  .  .  .  .  1  1  2  3
      23 |  .  .  .  .  .  .  .  .  .  .  .  .  .  .  1  3
      22 |  .  .  .  .  .  .  .  .  .  .  .  .  .  1  2  3
      21 |  .  .  .  .  .  .  .  .  .  .  .  .  1  1  2  5
      20 |  .  .  .  .  .  .  .  .  .  .  .  1  .  2  4  3
      19 |  .  .  .  .  .  .  .  .  .  .  .  .  1  1  2  3
      18 |  .  .  .  .  .  .  .  .  .  .  .  1  1  3  3  4
      17 |  .  .  .  .  .  .  .  .  .  .  .  .  1  2  3  4
      16 |  .  .  .  .  .  .  .  .  .  1  .  2  3  5  4
      15 |  .  .  .  .  .  .  .  .  .  1  1  3  2  3  6
      14 |  .  .  .  .  .  .  .  .  .  .  2  1  3  4  3
      13 |  .  .  .  .  .  .  .  .  .  .  2  3  3  3  2
      12 |  .  .  .  .  .  .  .  .  1  2  1  3  4  4  5
      11 |  .  .  .  .  .  .  .  .  .  1  2  3  2  3  2
      10 |  .  .  .  .  .  .  .  .  2  2  2  2  4  3  2
```

```
 9 |  .  .  .  .  .  .  1  1  1  3  5  4  4  4
 8 |  .  .  .  .  .  .  1  1  2  4  3  3  5  3
 7 |  .  .  .  .  .  .  1  1  4  3  4  3  3  4
 6 |  .  .  .  .  .  1  2  5  5  3  6  5  6  6
 5 |  .  .  .  .  .  1  4  2  3  4  5  4  6  5
 4 |  .  .  .  .  3  3  4  3  6  4  5  5  7  4
 3 |  .  .  .  .  3  3  3  4  5  4  8  5  5  5
 2 |  .  .  .  3  3  2  5  3  5  4  6  3  7  6
 1 |  .  1  3  2  3  3  5  3  5  4  5  5  8  5
 0 |  3  3  3  3  6  4  5  6  8  6  9  8  9 10
---+-----------------------------------------
g/d|  2  3  4  5  6  7  8  9 10 11 12 13 14 15
```

We close this section with a study of $\mathscr{H}^{\mathrm{sm}}_{8,5}$, the restricted Hilbert scheme of smooth curves of degree 8 and genus 5 in $\mathbb{P}^3$. It is known that $\mathscr{H}^{\mathrm{sm}}_{8,5}$ is irreducible; see for example [Ein 1986, Theorem 4]. In [Gruson and Peskine 1978, §4], we find a complete stratification of $\mathscr{H}^{\mathrm{sm}}_{8,5}$ into locally closed families A, B, C, D, E, as well as the Betti tables of each family. The following shows that the free resolutions for the families E, C, B, A (in this order) are indeed correct.

```
i13 : L = select(LC, C -> degree C == 8 and genus C == 5);
i14 : L / ideal / res / betti
             0 1 2 3         0 1 2 3         0 1 2 3         0 1 2 3
o14 = {total: 1 6 8 3, total: 1 5 5 1, total: 1 4 4 1, total: 1 7 8 2}
           0: 1 . . .      0: 1 . . .      0: 1 . . .      0: 1 . . .
           1: . 1 . .      1: . . . .      1: . . . .      1: . . . .
           2: . . . .      2: . 1 . .      2: . 1 . .      2: . . . .
           3: . . . .      3: . 3 3 .      3: . 3 2 .      3: . 7 8 2
           4: . . . .      4: . 1 2 1      4: . . 2 1
           5: . 5 8 3
o14 : List
```

We note that curves in family *D* are missing from our program because they do not lie on any of the surfaces we constructed. In fact, they all lie on the ruled cubic surface with a double line. In the upcoming 2.0 version of the package, we will include curves on this surface and much more.

3. ACM CURVES IN PROJECTIVE THREE-SPACE. In this section we consider arithmetically Cohen–Macaulay (ACM) curves in $\mathbb{P}^3$. The literature on ACM curves in $\mathbb{P}^3$ is very rich and we do not aim to give a survey in this article. We shall illustrate the stratification of Hilbert schemes of ACM curves by Betti tables.

3.1. ***Hilbert functions of ACM curves.*** Let $\mathscr{H}^{\mathrm{CM}}$ denote the points of the Hilbert scheme (without fixing a Hilbert polynomial) of curves in $\mathbb{P}^3$ that correspond to ACM curves. Ellingsrud [1975] shows that $\mathscr{H}^{\mathrm{CM}}$ is an open smooth subscheme of the Hilbert scheme. Let $\mathscr{H}^{\mathrm{CM}}_H$ denote the points that correspond to ACM curves

with Hilbert function H, then we have a stratification

$$\mathscr{H}^{\mathrm{CM}} = \bigsqcup_H \mathscr{H}^{\mathrm{CM}}_H.$$

Furthermore, it is shown in [Ellingsrud 1975] that the spaces $\mathscr{H}^{\mathrm{CM}}_H$ form disjoint irreducible connected components of $\mathscr{H}^{\mathrm{CM}}$.

What are all the possible Hilbert functions of ACM curves in $\mathbb{P}^3$?

There are many notions that encode information which is equivalent to that encoded by the Hilbert function, e.g., *numerical characters* as in [Gruson and Peskine 1978, §2], *postulation characters* as in [Martin-Deschamps and Perrin 1990, I.2] and *h-vectors* as in [Migliore 1998, §1.4]. For computational reasons that we will explain later, we will use the postulation character of a curve C, which is defined to be the negative of the third discrete difference of its Hilbert function.

A theorem of Gruson and Peskine [1978, §2] says that the postulation characters of ACM curves are exactly the *positive characters*. A positive character is a function $\gamma : \mathbb{Z} \to \mathbb{Z}$ of finite support, such that the following hold:

(1) $\sum_n \gamma(n) = 0$,

(2) $\gamma(n) = 0$ for $n < 0$,

(3) $\gamma(0) = -1$,

(4) If we set $s = s(\gamma) = \inf\{n | \gamma(n) \neq -1\}$, then $\gamma(s) \geq 0$,

(5) $\gamma(n) \geq 0$ for $n \geq s$.

The degree of a positive character γ is defined to be $\sum_n n\gamma(n)$. If γ is the postulation character of a curve C, then the degree of γ is the degree of C, and $s(\gamma)$ is the least degree surface C lies on.

Given d and s, the enumeration of all positive characters becomes a familiar problem: what are the different ways to make $d - \sum_{n<s} n$ cents in total using s coins where each coin can have value n for every $n \geq s$? Macaulay2 solves this efficiently by creating a bigraded ring and enumerating monomials of a fixed bidegree. The method function `positiveChars(ZZ,ZZ)` does exactly that, and enumerates all positive characters with a given degree d and s. By looping s from 1 to $d-1$, the method function `positiveChars(ZZ)` then enumerates all positive characters of a given degree d.

```
i2 : positiveChars(6,2)
o2 = {{-1, -1, 1, 0, 0, 1}, {-1, -1, 0, 1, 1}}
o2 : List
i3 : positiveChars(6)
o3 = {{-1, -1, 1, 0, 0, 1}, {-1, -1, 0, 1, 1}, {-1, -1, -1, 3}}
o3 : List
```

We can produce all 121 positive characters up to degree 16 within 0.05 seconds. The efficiency is one of the main reasons why we chose the postulation character

to represent the Hilbert function.

```
i4 : time L = flatten apply(15, d -> positiveChars(d+1));
     -- used 0.0492106 seconds
i5 : #L
o5 = 121
```

3.2. ***Betti tables of ACM curves.*** We define a further stratification

$$\mathscr{H}_H^{\mathrm{CM}} = \bigsqcup_B \mathscr{H}_{H,B}^{\mathrm{CM}},$$

where $\mathscr{H}_{H,B}^{\mathrm{CM}}$ consists of points that correspond to ACM curves with Hilbert function H and Betti table B.

What are all the Betti tables B of an ACM curve in $\mathbb{P}^3$?

A necessary and sufficient condition for a Betti table B of a homogeneous ideal to be that of an ACM curve in $\mathbb{P}^3$ is the following: B has $n+1$ generators of degrees $a_1 \geq \cdots \geq a_{n+1}$ and n syzygies of degrees $b_1 \geq \cdots \geq b_n$ and no higher syzygies, such that the diagonal entries of the matrix $M = (a_i - b_j)_{i,j}$ are positive. See, for example, [Eisenbud 2005, Propositions 3.8 and 3.14]. We call M the Hilbert–Burch degree matrix.

For a given Hilbert function H, let $\mathfrak{B}_H$ denote the set of Betti tables B such that $\mathscr{H}_{H,B}^{\mathrm{CM}}$ is nonempty. We partially order $\mathfrak{B}_H$ by the number of generators (equivalently syzygies) and write $B >_n B'$ if B has n more generators than B'. Since the Hilbert function H is fixed, the fourth discrete difference $\Delta^4 H$ is equal to the alternating sum of the Betti numbers $\sum_i (-1)^i B_{i,n}$ for every $B \in \mathfrak{B}_H$. Therefore $B >_1 B'$ if and only if B is obtained from B' by adding one generator and one syzygy of the same degree, and $B >_n B'$ if and only if B can be obtained from B' by n successive additions of one generator and one syzygy of the same degree. By the previous paragraph, if we remove a generator and a syzygy of the same degree from some $B \in \mathfrak{B}_H$ to obtain B', then $B' \in \mathfrak{B}_H$ also. Thus B is minimal in $\mathfrak{B}_H$ if and only if no generator and syzygy of B share the same degree, but in this case B is determined uniquely by $\Delta^4 H = \sum_i (-1)^i B_{i,n}$. In conclusion: $\mathfrak{B}_H$ has a smallest element denoted by $\mathfrak{b}$, and every $B \in \mathfrak{B}_H$ can be obtained from $\mathfrak{b}$ by successively adding one generator and one syzygy of the same degree (although not all ways of doing so yield a Betti table in $\mathfrak{B}_H$).

The main theorem of [Ellingsrud 1975] states that the downward closed strata $\bigsqcup_{B \leq B'} \mathscr{H}_{H,B}^{\mathrm{CM}}$ is open and irreducible (hence smooth) in $\mathscr{H}_H^{\mathrm{CM}}$. Further, for two Betti tables B, $B' \in \mathfrak{B}_H$, we have $B \geq B'$ if and only if $\mathscr{H}_{H,B'}^{\mathrm{CM}} \supset \mathscr{H}_{H,B}^{\mathrm{CM}}$. In other words, the partial order on $\mathfrak{B}_H$ corresponds to the partial order of specialization among the strata $\mathscr{H}_{H,B}^{\mathrm{CM}}$.

The method function `generalACMBetti(List)` takes a postulation character γ corresponding to the Hilbert function H and returns the smallest element $\mathfrak{b}$ of $\mathfrak{B}_H$.

```
i2 : generalACMBetti {-1,-1,-1,2,1}
            0 1 2
o2 = total: 1 3 2
         0: 1 . .
         1: . . .
         2: . 3 1
         3: . . 1
o2 : BettiTally
```

The method function `specializeACMBetti(BettiTally)` takes a Betti table $B \in B_H$ and returns the list of all $B' \in \mathfrak{B}_H$ such that $B' >_1 B$.

```
i3 : specializeACMBetti oo
             0 1 2
o3 = {total: 1 4 3}
          0: 1 . .
          1: . . .
          2: . 3 2
          3: . 1 1
o3 : List
```

Applying this iteratively on $\mathfrak{b}$, we exhaust all of $\mathfrak{B}_H$ in layers. The method function `allACMBetti(List)` takes a postulation character, and returns all Betti tables of ACM curves having that character. The following are all the Betti tables of ACM curves of degree 8. Each row corresponds to a distinct Hilbert function, and along each row the Betti tables are more special as we go from left to right. Note that these outputs are only Betti tables, no curves are produced yet.

```
i5 : netList (positiveChars(6) / allACMBetti)
     +------------+------------+
     |       0 1 2|            |
o5 = |total: 1 3 2|            |
     |    0: 1 . .|            |
     |    1: . 2 1|            |
     |    2: . . .|            |
     |    3: . . .|            |
     |    4: . 1 1|            |
     +------------+------------+
     |       0 1 2|       0 1 2|
     |total: 1 2 1|total: 1 3 2|
     |    0: 1 . .|    0: 1 . .|
     |    1: . 1 .|    1: . 1 .|
     |    2: . 1 .|    2: . 1 1|
     |    3: . . 1|    3: . 1 1|
     +------------+------------+
     |       0 1 2|            |
     |total: 1 4 3|            |
     |    0: 1 . .|            |
     |    1: . . .|            |
     |    2: . 4 3|            |
     +------------+------------+
```

The method function `degreeMatrix(BettiTally)` takes the Betti table of an ACM curve, and returns its Hilbert–Burch degree matrix. The method function `randomDeterminantalIdeal(Ring,Matrix)` computes the ideal of maximal minors of a matrix of random forms with specified degrees. Together, these method functions allow us to generate ACM curves. The following code generates an ACM curve for each of the Betti table given above, and shows that they do indeed have the predicted Betti tables.

```
i7 : L =  positiveChars(6) / allACMBetti;
i8 : netList apply(L, H -> H / (B ->
     betti res randomDeterminantalIdeal(ZZ/101[x,y,z,w],degreeMatrix B)))
     +------------+------------+
     |       0 1 2|            |
o8 = |total: 1 3 2|            |
     |    0: 1 . .|            |
     |    1: . 2 1|            |
     |    2: . . .|            |
     |    3: . . .|            |
     |    4: . 1 1|            |
     +------------+------------+
     |       0 1 2|       0 1 2|
     |total: 1 2 1|total: 1 3 2|
     |    0: 1 . .|    0: 1 . .|
     |    1: . 1 .|    1: . 1 .|
     |    2: . 1 .|    2: . 1 1|
     |    3: . . 1|    3: . 1 1|
     +------------+------------+
     |       0 1 2|            |
     |total: 1 4 3|            |
     |    0: 1 . .|            |
     |    1: . . .|            |
     |    2: . 4 3|            |
     +------------+------------+
```

Here is an example of two layers of specializations. The second and the third families are incomparable, and they both specialize to the fourth family.

```
i9 : gamma = (positiveChars(9))#5
o9 = {-1, -1, -1, 1, 1, 1}
o9 : List
i10 : allACMBetti gamma
              0 1 2         0 1 2         0 1 2         0 1 2
o10 = {total: 1 2 1, total: 1 3 2, total: 1 3 2, total: 1 4 3}
           0: 1 . .      0: 1 . .      0: 1 . .      0: 1 . .
           1: . . .      1: . . .      1: . . .      1: . . .
           2: . 2 .      2: . 2 1      2: . 2 .      2: . 2 1
           3: . . .      3: . 1 .      3: . . 1      3: . 1 1
           4: . . 1      4: . . 1      4: . 1 1      4: . 1 1
o10 : List
```

We end this section with a short comment on smoothness. When does $\mathscr{H}^{\mathrm{CM}}_{H,B}$ have a point that corresponds to a smooth ACM curve? The answer is given by [Geramita and Migliore 1989, Proposition 1.3]: if and only if the Hilbert–Burch degree matrix has positive upper diagonal. We leave it as an exercise to the reader to check which of the four families above have a smooth ACM curve.

4. MINIMAL CURVES IN AN EVEN LIAISON CLASS. The Hartshorne–Rao module (or deficiency module) of a curve C is defined to be the graded module

$$M_C := H^1_*(\mathscr{I}_C) = \bigoplus_{n\in\mathbb{Z}} H^0(\mathscr{I}_C(n))$$

over the polynomial ring $S = H^0_*(\mathscr{O}_{\mathbb{P}^3})$. Since the curves we consider are locally Cohen–Macaulay and equidimensional, one can show that M_C is of finite length.

The method `raoModule(Ideal)` computes the Hartshorne–Rao module from the ideal of a curve using local duality. We compute a free resolution $F_\bullet$ of S/I_C, then dualize and take the cokernel of the last term to compute $\mathrm{Ext}^3_S(S/I_C, S)$. Then we resolve $\mathrm{Ext}^3_S(S/I_C, S)$ by $G_\bullet$ and again dualize and take the cokernel of the last term to obtain M_C. The following is an example of the rational quartic curve.

```
ii78 : I = monomialCurveIdeal(R,{1,3,4});
oo78 : Ideal of R
ii79 : betti res I
             0 1 2 3
oo79 = total: 1 4 4 1
          0: 1 . . .
          1: . 1 . .
          2: . 3 4 1
oo79 : BettiTally
ii80 : M = raoModule I
oo80 = cokernel {1} | w -z y -x |
                               1
oo80 : R-module, quotient of R
ii81 : betti res M
             0 1 2 3 4
oo81 = total: 1 4 6 4 1
          1: 1 4 6 4 1
oo81 : BettiTally
```

Rao [1978/79] made the beautiful discovery that there is a bijection between the even liaison classes of curves and isomorphism classes of finite length modules up to a shift in grading. We say a curve C is minimal if for every curve D in the even liaison class of C, we have $M_D \cong M_C[h]$ for some $h \geq 0$. It turns out that a minimal curve also has the minimal degree and genus among all curves in its even liaison class, and they are unique up to deformation with constant cohomology. Furthermore, all curves in the even liaison class can be obtained from C by finitely many

basic double links and a deformation with constant cohomology. In particular, the possible degrees and genera of all curves in the even liaison class can be computed from the minimal curve C alone. This vastly generalizes the case of ACM curves where $H^1_*(\mathscr{I}_C) = 0$. We refer the readers to [Martin-Deschamps and Perrin 1990] for details.

Given a finite length graded S-module M, we wish to construct a minimal curve C such that $M_C \cong M[h]$ for some smallest possible integer h. A suitable algorithm is outlined in [Martin-Deschamps and Perrin 1990] and improved by [Guarrera et al. 1997]. We give a simplified account here. Consider the short exact sequence of S-modules

$$0 \to S/I_C \to H^0_*(\mathscr{I}_C) \to M_C \to 0.$$

If $F_\bullet$ and $G_\bullet$ are minimal free resolutions of S/I_C and $H^0_*(\mathscr{I}_C)$, respectively, and $\alpha : F_\bullet \to G_\bullet$ is an induced map, then the mapping cone $C(\alpha)_\bullet$ is a resolution of M_C. By [Martin-Deschamps and Perrin 1990, Theorems 3.7 and 4.1], if C is a minimal curve, then $C(\alpha)_\bullet$ is actually minimal. Since $H^0_*(\mathscr{I}_C)$ has depth 2, it has projective dimension 2 by the Auslander–Buchsbaum formula. We see that $F_2 \cong L_4$, $F_1 \cong L_3$ and F_0 is a summand of L_2.

We turn this observation around, and start with a minimal free resolution $L_\bullet$ of a finite length module M and ask: what are the conditions on the projections π such that $\operatorname{coker} d = \operatorname{coker}(\pi \circ d_3)$ is the ideal of a curve in $\mathbb{P}^3$ after shifting by an integer h?

The Buchsbaum–Eisenbud criterion of exactness tells us that if the lower complex in

$$\begin{array}{ccccccccc}
0 & \longrightarrow & L_4 & \longrightarrow & L_3 & \xrightarrow{d_3} & L_2 & \longrightarrow & \Omega^2 M \longrightarrow 0 \\
 & & \| & & \| & & \downarrow{\scriptstyle \pi} & & \downarrow \\
0 & \longrightarrow & L_4 & \longrightarrow & L_3 & \xrightarrow{d} & P & \longrightarrow & \operatorname{coker} d \longrightarrow 0
\end{array}$$

is a free resolution of an ideal (up to shift), then $\operatorname{rk} P = \operatorname{rk} L_3 + 1 - \operatorname{rk} L_4 = \operatorname{rk} d + 1$ and grade $I(d) \geq 2$. It turns out that this is also sufficient. We rephrase the grade $I(d) \geq 2$ condition in terms of the ideal of maximal nonvanishing minors: this is if and only if $I(d)$ is not contained in any principal ideal, i.e., $\operatorname{rk} d \otimes_S S/(f) = \operatorname{rk} d$ for all $f \in S$. Since the alternating sum of the degrees of the free modules must be zero for the free resolution of a curve by the Herzog–Kühl equations, we can easily determine h whenever the degrees of P are known. To summarize, we are looking for projections π satisfying the above conditions such that the resulting shift h is as small as possible.

The key observation of [Guarrera et al. 1997] is that the rank and the rank in codimension 1 of a matrix M over a PID can be easily read off from its Smith normal form. Furthermore, if we evaluate the matrix d in a PID $k[T]$, then for a general

choice of evaluations $k[x_0, \ldots, x_3] \to k[t]$, the rank and rank in codimension 1 of d are preserved. The method function `minimalCurve(Module)` implements this method to find a projection π corresponding to a minimal curve.

Let us generate a minimal curve in the even liaison class of the rational quartic.

```
ii82 : J = minimalCurve M;
oo82 : Ideal of R
ii83 : betti res J
             0 1 2 3
oo83 = total: 1 4 4 1
          0: 1 . . .
          1: . 4 4 1
oo83 : BettiTally
```

Indeed, the rational quartic curve has Hartshorne–Rao module k concentrated in degree 1, and lies on a smooth quadric with three other cubic generators. The complete intersection of the quadric with a general linear combination of the cubic generators is a divisor of type $(3, 3)$ on the quadric surface. Since the rational quartic is of type $(1, 3)$, the residual would have type $(2, 0)$ — the union of two skew lines. The latter has Hartshorne–Rao module k concentrated in degree 0, and is obviously of minimal degree in its even liaison class.

Finally, here is a remark about the function `minimalCurve(Module)`. Once the matrix d is computed, there are two ways to obtain the equations of the ideal I_C: the first is to compute the kernel of the transposition of d, the second is to take maximal minors and saturate by the irrelevant ideal. Unfortunately, both would take a long time if d is a large matrix. To get partial information, the method function `minimalCurveBetti(Module)` outputs the Betti table of a minimal curve corresponding to a finite length module without explicitly computing the ideal of the curve.

ACKNOWLEDGEMENTS. The author thanks F. Schreyer and M. Stillman for initiating the project and R. Hartshorne for guidance. The author also thanks D. Eisenbud for supervision and support.

SUPPLEMENT. The online supplement contains version 1.0 of SpaceCurves.

REFERENCES.

[Ein 1986] L. Ein, "Hilbert scheme of smooth space curves", *Ann. Sci. École Norm. Sup.* (4) **19**:4 (1986), 469–478. MR Zbl

[Eisenbud 2005] D. Eisenbud, *The geometry of syzygies: a second course in commutative algebra and algebraic geometry*, Graduate Texts in Mathematics **229**, Springer, 2005. MR Zbl

[Ellingsrud 1975] G. Ellingsrud, "Sur le schéma de Hilbert des variétés de codimension 2 dans $\mathbf{P}^e$ à cône de Cohen–Macaulay", *Ann. Sci. École Norm. Sup.* (4) **8**:4 (1975), 423–431. MR Zbl

[Geramita and Migliore 1989] A. V. Geramita and J. C. Migliore, "Hyperplane sections of a smooth curve in $\mathbf{P}^3$", *Comm. Algebra* **17**:12 (1989), 3129–3164. MR Zbl

[Gruson and Peskine 1978] L. Gruson and C. Peskine, "Genre des courbes de l'espace projectif", pp. 31–59 in *Algebraic geometry* (Tromsø, 1977), edited by L. D. Olsen, Lecture Notes in Math. **687**, Springer, Berlin, 1978. MR Zbl

[Gruson and Peskine 1982] L. Gruson and C. Peskine, "Genre des courbes de l'espace projectif, II", *Ann. Sci. École Norm. Sup.* (4) **15**:3 (1982), 401–418. MR Zbl

[Guarrera et al. 1997] S. Guarrera, A. Logar, and E. Mezzetti, "An algorithm for computing minimal curves", *Arch. Math.* (*Basel*) **68**:4 (1997), 285–296. MR Zbl

[Halphen 1882] G. Halphen, "Mémoire sur la classification des courbes gauches algébriques", *J. Éc. Poly.* **52** (1882), 1–200.

[Hartshorne 1977] R. Hartshorne, *Algebraic geometry*, Graduate Texts in Mathematics **52**, Springer, 1977. MR Zbl

[Hartshorne 1982] R. Hartshorne, "Genre de courbes algébriques dans l'espace projectif (d'après L. Gruson et C. Peskine)", exposé 592] 301–313 in *Séminaire Bourbaki,* 1981/1982, Astérisque, Soc. Math. France, Paris, 1982. MR Zbl

[Martin-Deschamps and Perrin 1990] M. Martin-Deschamps and D. Perrin, *Sur la classification des courbes gauches*, Astérisque **184-185**, 1990. MR Zbl

[Migliore 1998] J. C. Migliore, *Introduction to liaison theory and deficiency modules*, Progress in Mathematics **165**, Birkhäuser, Boston, 1998. MR Zbl

[Rao 1978/79] P. Rao, "Liaison among curves in $\mathbf{P}^3$", *Invent. Math.* **50**:3 (1978/79), 205–217. MR Zbl

RECEIVED: 19 Oct 2017 REVISED: 10 Apr 2018 ACCEPTED: 18 May 2018

MENGYUAN ZHANG:

myzhang@berkeley.edu

Department of Mathematics, University of California, Berkeley, United States

JSAG **8** (2018), 49–60
dx.doi.org/10.2140/jsag.2018.8.49

The Journal of Software for
Algebra and Geometry

The ReesAlgebra package in Macaulay2

DAVID EISENBUD

ABSTRACT: This note introduces Rees algebras and some of their uses, with illustrations from version 2.2 of the Macaulay2 package `ReesAlgebra.m2`.

INTRODUCTION. A central construction in modern commutative algebra starts from an ideal I in a commutative ring R, and produces the *Rees algebra*

$$\mathcal{R}(I) := R \oplus I \oplus I^2 \oplus I^3 \oplus \cdots \cong R[It] \subset R[t],$$

where $R[t]$ denotes the polynomial algebra in one variable t over R. For basics on Rees algebras, see [Vasconcelos 1994] and [Swanson and Huneke 2006], and for some other research, see [Eisenbud and Ulrich 2018; Kustin and Ulrich 1992; Ulrich 1994], and [Valabrega and Valla 1978].

From the point of view of algebraic geometry, the Rees algebra $\mathcal{R}(I)$ is a homogeneous coordinate ring for the graph of a rational map whose total space is the blowup of Spec R along the scheme defined by I. (In fact, the "Rees algebra" is sometimes called the "blowup algebra".)

Rees algebras were first studied in the algebraic context by David Rees, in the now-famous paper [Rees 1958]. Actually, Rees mainly studied the ring $R[It, t^{-1}]$, now also called the *extended Rees algebra* of I.

Mike Stillman and I wrote a Rees algebra script for Macaulay classic. It was augmented, and made into the [Macaulay2] package `ReesAlgebra.m2` around 2002, to study a generalization of Rees algebras to modules described in [Eisenbud et al. 2003]. Subsequently Amelia Taylor, Sorin Popescu, the present author, and, at the Macaulay2 Workgroup in July 2017, Ilir Dema, Whitney Liske, and Zhangchi Chen contributed routines for computing many of the invariants of an ideal or module defined in terms of Rees algebras. These routines comprise the package's primary utility, since Rees algebras of modules other than ideals are comparatively little studied.

The author is grateful to the National Science Foundation for partial support.
MSC2010: primary 13A30, 13B22, 13D02; secondary 14C17, 14E15.
Keywords: Rees algebras, blowup, distinguished subvariety, special fiber.
ReesAlgebra.m2 version 2.2

We first describe the construction and an example from [Eisenbud et al. 2003]. Then we list some of the functionality the package now has and illustrate it with a theorem of Morey and Ulrich. Finally we give examples of how Rees algebras appear in the Fulton–MacPherson intersection theory and in the resolution of singularities.

1. THE REES ALGEBRA OF A MODULE. There are several possible ways of extending the Rees algebra construction from ideals to modules. For simplicity we will henceforward only consider finitely generated modules over Noetherian rings. Huneke and Ulrich and I argued in [Eisenbud et al. 2003] that the most natural way to extend the definition is to think of $R[It]$ as the image of the map of symmetric algebras $\mathrm{Sym}(\phi) : \mathrm{Sym}_R(I) \to \mathrm{Sym}_R(R) = R[t]$, and to generalize it to the case of an arbitrary finitely generated module M by setting

$$\mathcal{R}(M) = \text{image } \mathrm{Sym}(\phi),$$

where ϕ is a *versal* map from M to a free module. Such a versal map may be computed as the composition of the diagonal embedding

$$M \to \oplus_{i=1}^{m} M,$$

with the map

$$\oplus_{i=1}^{m} \phi_i : \oplus_{i=1}^{m} M \to R^m,$$

where $\phi_1, \ldots, \phi_m$ generate $\mathrm{Hom}_R(M, R)$.

Though this is not immediate, the Rees algebra of an ideal in a Noetherian ring, in this sense, is the same as the Rees algebra in the classical sense, and in most cases one can take any embedding of the module into a free module in the definition:

Theorem 1.1 [Eisenbud et al. 2003, Theorems 0.2 and 1.4]. *Let R be a Noetherian ring and let M be a finitely generated R-module. Let $\phi : M \to G$ be a versal map of M to a free module. Suppose that ϕ is an inclusion, and let $\psi : M \to G'$ be any inclusion of M into a free module G'. If R is torsion-free over $\mathbb{Z}$ or R is unmixed and generically Gorenstein or M is free locally at each associated prime of R, or $G' = R$, then the image of* $\mathrm{Sym}(\phi)$ *and the image of* $\mathrm{Sym}(\psi)$ *are naturally isomorphic.*

Nevertheless some examples do violate the conclusion of Theorem 1.1. Here is one from [Eisenbud et al. 2003] in characteristic 5 (any finite characteristic would work similarly).

```
i1 : p = 5;
i2 : R = ZZ/p[x,y,z]/(ideal(x^p,y^p)+(ideal(x,y,z))^(p+1));
i3 : M = module ideal(z);
```

It is easy to check that $M \cong R^1/(x, y, z)^p$. We write $\iota : M \to R^1$ for the embedding as an ideal and ψ for the embedding $M \to R^2$ sending z to the vector (x, y).

```
i4 : iota = map(R^1,M,matrix{{z}});
i5 : psi = map(R^2,M,matrix{{x},{y}});
```

Finally, we choose a versal embedding $M \to R^3$. It sends z to the vector (x, y, z):

```
i6 : phi = versalEmbedding(M);
```

We now compute the kernels of the three maps on symmetric algebras:

```
i7 : Iiota = symmetricKernel iota;
i8 : Ipsi = symmetricKernel psi;
i9 : Iphi = symmetricKernel phi;
```

and check that the ones corresponding to ϕ and ι are equal, whereas the ones corresponding to ψ and ϕ are not — they differ in degree p.

```
i10 : Iiota == Iphi
o10 = true
i11 : Ipsi == Iphi
o11 = false
i12 : numcols basis(p,Iphi)
o12 = 3
i13 : numcols basis(p,Ipsi)
o13 = 1
```

2. THE REES ALGEBRA AND ITS RELATIONS. The central routine, `reesIdeal` (with synonym: `reesAlgebraIdeal`), computes an ideal defining the Rees algebra $\mathcal{R}(M)$ as a quotient of a polynomial ring over R from a free presentation of M. From the Rees ideal we immediately get `reesAlgebra` M. In the case when M is an ideal in R we also compute the important `associatedGradedRing` $M = \mathcal{R}(M)/M$ (and the more geometric sounding but identical `normalCone` M). If I is a (homogeneous) ideal primary to the maximal ideal of a standard graded ring R we compute the Hilbert–Samuel multiplicity of I with the routine `multiplicity`.

We now describe the basic computation. Suppose that M has a set of generators represented by a map from a free module,

$$F \xrightarrow{\alpha} M \to 0,$$

and suppose $F = R^n$. The symmetric algebra of F over R is then a polynomial ring $\mathrm{Sym}_R(F) = R[t_1, \ldots, t_n]$ on n new indeterminates $t_1, \ldots, t_n$. By the universal property of the symmetric algebra there is a canonical surjection $\mathrm{Sym}_R(F) \to \mathrm{Sym}_R(M)$, so we may compute the Rees algebra of M as a quotient of $\mathrm{Sym}_R(F)$. The expression

$$I = \texttt{reesIdeal}\ M$$

first uses `versalEmbedding` M to compute a versal map from M to a free module $\beta : M \to G$. The expression `symmetricKernel` $\alpha \circ \beta$ then constructs the map of

symmetric algebras $\beta \circ \alpha : \mathrm{Sym}_R(F) \to \mathrm{Sym}_R(G)$ and uses the built-in Macaulay2 routine to compute the kernel

$$I = \texttt{reesIdeal}\ M = \ker \mathrm{Sym}(\beta \circ \alpha) : \mathrm{Sym}_R(F) \to \mathrm{Sym}_R(G).$$

There is a different way of computing the Rees algebra that is often much more efficient. It begins by constructing the symmetric algebra of M, and uses the observation that the construction of the Rees algebra commutes with localization. See [Eisenbud 1995, Appendix 2] for the necessary facts about symmetric algebras.

Suppose that M has a free presentation,

$$G \xrightarrow{\phi} F \xrightarrow{\alpha} M \to 0.$$

The right exactness of the symmetric algebra functor implies that the symmetric algebra of M is the quotient of $\mathrm{Sym}_R(F)$ by an ideal I_0 that is generated by the entries of the matrix

$$(t_1 \ \cdots \ t_n) \circ \phi,$$

(where we have identified ϕ with $\mathrm{Sym}_R(F) \otimes_R \phi$). Thus I_0 is generated by polynomials that are linear in the variables t_i (and because M is the degree 1 part of $\mathcal{R}(M)$, these are the only linear forms in the t_i in the Rees ideal).

If $f \in R$ is an element such that $M[f^{-1}]$ is free on generators $g_1, \ldots, g_n$, it follows that after inverting f, the Rees algebra of M becomes a polynomial ring over $R[f^{-1}]$ on indeterminates corresponding to the g_i:

$$\mathcal{R}(M)[f^{-1}] = \mathrm{Sym}_R(M[f^{-1}]) = R[G_1, \ldots, G_n].$$

Now suppose in addition that f is a non-zerodivisor in R. In the diagram

$$\begin{array}{ccccc}
\mathrm{Sym}_R(F) & \xrightarrow{\alpha} & \mathrm{Sym}_R(M) & \xrightarrow{\beta} & \mathrm{Sym}_R(G) \\
\downarrow & & \downarrow & & \downarrow \\
\mathrm{Sym}_R(F)[f^{-1}] & \xrightarrow{\alpha} & \mathrm{Sym}_R(M)[f^{-1}] & \xrightarrow{\beta} & \mathrm{Sym}_R(G)[f^{-1}]
\end{array}$$

the two outer vertical maps are inclusions, and it follows that the Rees ideal, which is the kernel of the map $\mathcal{R}(F) = \mathrm{Sym}_R(F) \to \mathcal{R}(M)$, is equal to the intersection of $\mathcal{R}(F)$ with the kernel of

$$\mathrm{Sym}_R(F)[f^{-1}] \xrightarrow{\beta} \mathrm{Sym}_R(G)[f^{-1}].$$

This intersection may be computed as $I_0 : f^\infty$. The command

$$\texttt{reesIdeal}(I, f)$$

computes the Rees ideal in this way.

More generally, we say that a module N is *of linear type* if the Rees ideal of M is equal to the ideal of the symmetric algebra of M; for example, any complete

intersection ideal is of linear type, and the condition can be tested by the command

`isLinearType` M.

The procedure above really requires only that f be a non-zerodivisor in R and that $M[f^{-1}]$ be of linear type over $R[f^{-1}]$.

3. REDUCTIONS AND THE SPECIAL FIBER. A *reduction* J of an ideal I is a subideal $J \subset I$ over which I is *integrally dependent*. In concrete terms this means that there is some integer r such that $JI^r = I^{r+1}$, and the minimal r with this property is called the reduction number. The property of being a reduction is tested by `isReduction` I, and `reductionNumber` I computes the reduction number.

Now suppose that $\mathfrak{m}$ is a maximal ideal containing I. The special fiber ring is by definition $\mathcal{R}(I)/\mathfrak{m}\mathcal{R}(I)$. It is a standard graded algebra over the field $k := R/\mathfrak{m}$, a quotient of $\mathrm{Sym}_R(F)/\mathfrak{m} = k[t_1, \ldots, t_n]$ where, as before, F is a free module of rank n with a surjection to M. The defining ideal of the special fiber ring, and the ring itself, are computed using `specialFiberIdeal` I and `specialFiberRing` I.

The dimension of the special fiber ring is called the analytic spread of I, usually denoted

$$\ell(I) = \texttt{analyticSpread}\ I.$$

Northcott and Rees [1954] proved that if k is infinite then there always exist reductions generated by $\ell(I)$ elements, and this is the minimum possible number; these are called minimal reductions. The smallest possible reduction number for I with respect to a minimal reduction is by definition `reductionNumber` I. (This is always achieved by any ideal generated by $\ell(I)$ sufficiently general scalar linear combinations of the generators of I; but note that when I is homogeneous but has generators of different degrees such linear combinations are sometimes necessarily inhomogeneous.)

An interesting special case occurs when R is a graded ring over $k = R_0$ and the generators $g_1, \ldots, g_n$ of I are all homogeneous of the same degree. In this case the special fiber ring is easily seen to be equal to the subring $k[g_1, \ldots, g_n]$ (usually *not* a polynomial ring) generated by the elements g_i.

For example, if I is the ideal of $p \times p$ minors of a $p \times (p+q)$ matrix, then the special fiber ring is equal to the homogeneous coordinate ring $\mathbb{G}$ of the Grassmannian of p-planes in $p+q$ space. It follows that $\ell(I) = \dim \mathbb{G} = pq + 1$, and the reduction number of I is $(p-1)(q-1)$.

4. FINDING ELEMENTS OF THE REES IDEAL. Let M be an R-module and let $\phi : R^s \to R^m$ be its presentation matrix. We identify $\mathrm{Sym}_R(R^m)$ with the polynomial ring $R[t_1, \ldots, t_m]$. By the universality of the symmetric algebra construction, the

symmetric algebra of I has the form

$$\mathrm{Sym}_R(I) = R[t_1, \ldots, t_m]/(T\phi),$$

where we have written T for the vector $(t_1 \ \ldots \ t_m) \in R[t_1, \ldots, t_m]^m$, whose entries correspond to the generators of I, and written $(T\phi)$ for the ideal generated by the entries of the product

$$(t_1 \ \cdots \ t_m)\phi.$$

If $J := (x_1, \ldots, x_n) \subset R$ is an ideal containing I, and we write

$$X = (x_1 \ \cdots \ x_n) \in R[t_1, \ldots, t_m]^n,$$

then there is a matrix ψ defined over $R[t_1, \ldots, t_m]$, called the Jacobian dual of ϕ with respect to X, such that $T\phi = X\psi$. (The matrix ψ is generally not unique; Macaulay2 computes it using Gröbner division with remainder.)

If I, J each contain a non-zerodivisor then J will have grade ≥ 1 on the Rees algebra $\mathcal{R}(I)$. Since $(T\phi)$ is contained in the defining ideal of the Rees algebra, the vector X is annihilated by the matrix ψ when regarded over the Rees algebra, and the relation $X\psi \equiv 0$ in $\mathcal{R}(I)$ implies that the $m \times m$ minors of ψ are in the Rees ideal of I.

In very favorable circumstances, one may even have the equality

$$\texttt{reesIdeal}\, I == \texttt{ideal}(T\phi) + \texttt{minors}(m, \psi).$$

We illustrate with a theorem of Morey and Ulrich. Recall that an ideal I is said to satisfy the condition G_ℓ if the number of generators of the localized ideal I_P is $\leq \operatorname{codim} P$ for every prime ideal P of codimension $< \ell$; equivalently, if I has presentation matrix ϕ as above,

$$\operatorname{codim} I_{m-p}(\phi) > p$$

for $1 \leq p < \ell$.

Theorem 4.1 [Morey and Ulrich 1996]. *Let R be a local Gorenstein ring with infinite residue field, let I be a perfect ideal of grade 2 with m generators, let ϕ be the presentation matrix of I, and let ψ be the Jacobian dual matrix. Let $\ell = \ell(I)$ be the analytic spread. Suppose that I satisfies the condition G_ℓ. The following conditions are equivalent*:

(1) *$\mathcal{R}(I)$ is Cohen–Macaulay and $I_{(m-\ell)}(\phi) = I_1(\phi)^{m-\ell}$.*

(2) *$r(I) < \ell$ and $I_{m+1-\ell}\phi = (I_1\phi)^{m+1-\ell}$.*

(3) *The ideal of $\mathcal{R}(I)$ is equal to the sum of the ideal of* $\mathrm{Sym}(I)$ *with the Jacobian dual minors, $I_m\psi$.*

We can check all these conditions with functions in the package. We start with the presentation matrix ϕ of an $m{=}n{+}1$-generator perfect ideal such that the first

row consists of the n variables of the ring, and the rest of the rows are reasonably general (in this case random quadrics):

```
i2 : setRandomSeed 0
i3 : n=3;
i4 : kk = ZZ/101;
i5 : S = kk[a_0..a_(n-2)];
i6 : phi = transpose map(S^(n-1),S^{-1,(n-1):-2},
             (i,j) -> if j == 0 then a_i else random(2,S));
             3       2
o6 : Matrix S  <--- S
i7 : I = minors(n-1,phi);
```

This is a perfect codimension 2 ideal, as we see from the Betti table:

```
i8 : betti (F = res I)
            0 1 2
o8 = total: 1 3 2
         0: 1 . .
         1: . . .
         2: . 2 .
         3: . 1 2
```

We compute the analytic spread ℓ and the reduction number r:

```
i12 : ell = analyticSpread I
o12 = 2
i13 : r = reductionNumber(I, minimalReduction I)
o13 = 1
```

Now we can check the condition G_ℓ, first probabilistically:

```
i15 : whichGm I >= ell
o15 = true
```

and now deterministically:

```
i17 : apply(toList(1..ell-1),
                  p-> {p+1, codim minors(n-p, phi)})
o17 = {{2, 2}}
```

We now check the three equivalent conditions of the Morey–Ulrich theorem. Since $\ell = n-1$ in this case, the second parts of conditions (1) and (2) are vacuously satisfied, and since $r < \ell$ the conditions must all be satisfied. We first check that $\mathcal{R}(I)$ is Cohen–Macaulay:

```
i19 : reesI = reesIdeal I;
o19 : Ideal of S[w , w , w ]
                  0   1   2
i20 : codim reesI
o20 = 2
```

```
i21 : betti res reesI
            0 1 2
o21 = total: 1 3 2
          0: 1 . .
          1: . . .
          2: . 2 .
          3: . 1 2
```

Finally, we wish to see that `reesIdeal` I is generated by the ideal of the symmetric algebra together with the Jacobian dual:

```
i23 : psi = jacobianDual phi;
                             2                     2
o23 : Matrix (S[w , w , w ])  <--- (S[w , w , w ])
                0   1   2             0   1   2
```

We now compute the ideal J of the symmetric algebra; we do this by hand, since the command `symmetricAlgebra` I would return the ideal over a different ring.

```
i25 : ST = ring psi
i26 : T = vars ST
o26 = | w_0 w_1 w_2 |
i27 : J = ideal(T*promote(phi, ST))
i28 :      betti res J
             0 1 2
o28 = total: 1 2 1
          0: 1 . .
          1: . . .
          2: . 2 .
          3: . . .
          4: . . 1
i29 : J1 = minors(ell, psi)
```

We compute the resolution of $G := J + J1$, to see that the resulting ideal is perfect, which also shows that it is the full ideal of the Rees algebra. We also check directly that it has the same resolution as the computed Rees ideal of I:

```
i30 : betti (G = res trim (J+J1))
            0 1 2
o30 = total: 1 3 2
          0: 1 . .
          1: . . .
          2: . 2 .
          3: . 1 2
i31 : betti res reesIdeal I
            0 1 2
o31 = total: 1 3 2
          0: 1 . .
          1: . . .
          2: . 2 .
          3: . 1 2
```

5. DISTINGUISHED SUBVARIETIES. The key construction in the Fulton–MacPherson definition of the refined intersection product [Fulton 1998, Section 6.1] involves normal cones, and is easy to implement using the tools in this package. The simplest case is the intersection of two subvarieties $X, V \subset Y$. If X and V meet in the *expected dimension*, defined to be $\dim V - \operatorname{codim}_Y X$, and the ambient variety Y is smooth, then one can assign multiplicities m_i to the components W_i of $X \cap V$, and the intersection product has the form $[X][V] = \sum m_i[W_i]$. The astonishing result of the Fulton–MacPherson theory is that if $X \subset Y$ is locally a complete intersection, then, no matter how singular Y and no matter how strange the actual intersection $X \cap V$, the intersection product $X \cdot V$ can be given a meaning as a rational equivalence class of cycles of the expected dimension on X, or even on certain *distinguished* subvarieties Z_i of $X \cap V$. This class comes with a canonical decomposition $\sum_i m_i \alpha_i$, where the m_i are positive integers, and α_i is a cycle of the expected dimension (possibly 0) on $Z_i \subset X \cap V$ (the same Z_i can appear several times, with different multiplicities and cycles).

In the general case, the subvariety V is replaced by a morphism $f : V \to Y$ from a variety V, and this is the key to the functoriality of the intersection product. The routines in this package work in the general setting, but for simplicity we will stick with the basic case in this description.

We now describe the distinguished subvarieties and their multiplicities. This part of the construction sheafifies, so (as in the package) we work in the affine case. We do not require any hypothesis on X, Y or V.

Let S be a ring (for example, the coordinate ring of Y) and let $I \subset S$ be an ideal (for example, the ideal of X). Write

$$T := \operatorname{gr}_I S = S/I \oplus I/I^2 \oplus \cdots$$

for the associated graded ring of I, and let π be the inclusion of S/I into T as the degree 0 part.

Let $f : S \to R$ be a ring homomorphism (for example, representing the projection $S \to S/(I(V))$). Let $K \subset T$ be the kernel of the induced map $\operatorname{gr}_I S \to \operatorname{gr}_{f(I)R} R$.

Let $P_1, \ldots, P_m$ be the minimal primes over K in $\operatorname{gr}_I R$. We define p_i to be the degree 0 part of P_i; that is, $p_i := P_i \cap S/I$. These are the distinguished prime ideals of S/I, and they clearly contain the kernel of $\overline{f} : S/I \to R/f(I)R$, so in the case where $R = S/J$ they contain $I + J$. Thus, in this case, they represent subvarieties of $X \cap V$.

Let m_i be the multiplicity with which P_i appears in the primary decomposition of K — that is,

$$m_i := \operatorname{length}_{\kappa(P_i)} P_{i\,P_i}/K_{P_i},$$

where $\kappa(P_i) = T_{P_i}/P_{i_{P_i}}$ is the residue field at P_i. Returning to geometric language, and the case where $X \subset Y$ is locally a complete intersection in a quasiprojective

variety, the cycle class α_i in the Chow group of the variety Z_i corresponding to p_i is defined as the Gysin image of the class of the subvariety corresponding to P_i in the projectivized normal bundle of X in Y — a construction not included in this package.

Here are some simple examples in which `distinguished` is used to compute the distinguished varieties of intersections in $\mathbb{A}^n$, via the function `intersectInP`. First, the familiar multiplicity 2 intersection of a conic with a tangent line.

```
i2 : kk = ZZ/101;
i3 : P = kk[x,y];
i4 : I = ideal"x2-y";J=ideal y;
i6 : intersectInP(I,J)
o6 = {{2, ideal (y, x)}}
```

Slightly more interesting, the following shows what happens when the intersections aren't rational:

```
i7 : I = ideal"x4+y3+1";
i8 : intersectInP(I,J)
                        2                          2
o8 = {{1, ideal (y, x  + 10)}, {1, ideal (y, x  - 10)}}
```

The real interest in the construction is in the case of improper intersections. Here are some typical results:

```
i9 : I = ideal"x2y";J=ideal"xy2";
i11 : intersectInP(I,J)
o11 = {{2, ideal x}, {5, ideal (y, x)}, {2, ideal y}}
i12 : intersectInP(I,I)
o12 = {{1, ideal y}, {4, ideal x}, {4, ideal (y, x)}}
```

6. REES ALGEBRAS AND DESINGULARIZATION. We conclude with an example illustrating a general result about projective birational maps of varieties. Recall that a map $B \to X$ of varieties is projective if it is the composition of a closed embedding $B \subset X \times \mathbb{P}^n$ with the projection to X. It is birational if it is generically an isomorphism. The inclusion of a ring into the Rees algebra of an ideal corresponds to a map from Proj of the Rees algebra to Spec of the ring, called a blowup, that is such a proper birational transformation, and in fact every proper birational transformation to an affine variety (or more generally to any scheme, if one works with sheaves of ideals) can be realized in this way.

The theorem of embedded resolution of singularities (proven by Hironaka in characteristic 0 and conjectured in general) says that, given any subvariety X of a smooth variety Y, there is a finite sequence of blowups

$$B_n \to \cdots \to B_2 \to B_1 \to Y$$

of smooth subvarieties that lie over the singular set of X, and a component of the preimage of X in B_n that is smooth and maps birationally to X. In the case of

plane curves, this can be done with a sequence of blowups of closed points. But in fact *any* sequence of blowups of a quasiprojective variety can be replaced with a single blowup [Hartshorne 1977, Theorem II.7.17] of a more complicated ideal. We illustrate this with the desingularization of a tacnode (the union of two smooth curves that meet with a simple tangency).

Example 6.1. Blowing-up (x^2, y) in $k[x, y]$ desingularizes the tacnode $x^2 - y^4$ in a single step.

```
i1 : R = ZZ/32003[x,y];
i2 : tacnode = ideal(x^2-y^4);
i3 : mm = ideal(x,y^2);
i4 : B = first flattenRing reesAlgebra mm;
i5 : irrelB = ideal(w_0,w_1);
i6 : proj = map(B,R,{x,y});
i7 : totalTransform = proj tacnode
              4    2
o7 = ideal(- y  + x )
i8 : netList (D = decompose totalTransform)
     +-----------------------+
o8 = |ideal (y, x)           |
     +-----------------------+
     |        2              |
     |ideal (y  + x, w  + w )|
     |                0    1 |
     +-----------------------+
     |        2              |
     |ideal (y  - x, w  - w )|
     |                0    1 |
     +-----------------------+
i9 : exceptional = proj mm
                2
o9 = ideal (x, y )
i10 : strictTransform = saturate(
                  totalTransform, exceptional);

i11 : netList decompose strictTransform
      +-----------------------+
      |        2              |
o11 = |ideal (y  + x, w  + w )|
      |                0    1 |
      +-----------------------+
      |        2              |
      |ideal (y  - x, w  - w )|
      |                0    1 |
      +-----------------------+
i12 : sing0 = sub(ideal singularLocus strictTransform, B);
i13 : sing = saturate(sing0,irrelB)
o13 = ideal 1
```

The last line asserts that the singular locus of the strict transform is empty; that is, the scheme defined by `strictTransform` is smooth (in this case it is the union of two disjoint smooth curves).

SUPPLEMENT. The online supplement contains version 2.2 of `ReesAlgebra.m2`.

REFERENCES.

[Eisenbud 1995] D. Eisenbud, *Commutative algebra with a view toward algebraic geometry*, Graduate Texts in Mathematics **150**, Springer, 1995. MR Zbl

[Eisenbud and Ulrich 2018] D. Eisenbud and B. Ulrich, "Duality and socle generators for residual intersections", *Journal für die reine and angewandte Mathematik* (online publication July 2018).

[Eisenbud et al. 2003] D. Eisenbud, C. Huneke, and B. Ulrich, "What is the Rees algebra of a module?", *Proc. Amer. Math. Soc.* **131**:3 (2003), 701–708. MR Zbl

[Fulton 1998] W. Fulton, *Intersection theory*, 2nd ed., Ergebnisse der Mathematik (3) **2**, Springer, 1998. MR Zbl

[Hartshorne 1977] R. Hartshorne, *Algebraic geometry*, Graduate Texts in Mathematics **52**, Springer, 1977. MR Zbl

[Kustin and Ulrich 1992] A. R. Kustin and B. Ulrich, "A family of complexes associated to an almost alternating map, with applications to residual intersections", *Mem. Amer. Math. Soc.* **95**:461 (1992), iv+94. MR

[Macaulay2] D. R. Grayson and M. E. Stillman, "Macaulay2: a software system for research in algebraic geometry", available at http://www.math.uiuc.edu/Macaulay2.

[Morey and Ulrich 1996] S. Morey and B. Ulrich, "Rees algebras of ideals with low codimension", *Proc. Amer. Math. Soc.* **124**:12 (1996), 3653–3661. MR Zbl

[Northcott and Rees 1954] D. G. Northcott and D. Rees, "Reductions of ideals in local rings", *Proc. Cambridge Philos. Soc.* **50** (1954), 145–158. MR Zbl

[Rees 1958] D. Rees, "On a problem of Zariski", *Illinois J. Math.* **2** (1958), 145–149. MR Zbl

[Swanson and Huneke 2006] C. Huneke and I. Swanson, *Integral closure of ideals, rings, and modules*, London Mathematical Society Lecture Note Series **336**, Cambridge University Press, 2006. MR Zbl

[Ulrich 1994] B. Ulrich, "Artin-Nagata properties and reductions of ideals", pp. 373–400 in *Commutative algebra: syzygies, multiplicities, and birational algebra (South Hadley, MA, 1992)*, edited by W. J. Heinzer et al., Contemp. Math. **159**, Amer. Math. Soc., Providence, RI, 1994. MR Zbl

[Valabrega and Valla 1978] P. Valabrega and G. Valla, "Form rings and regular sequences", *Nagoya Math. J.* **72** (1978), 93–101. MR

[Vasconcelos 1994] W. V. Vasconcelos, *Arithmetic of blowup algebras*, London Mathematical Society Lecture Note Series **195**, Cambridge University Press, 1994. MR Zbl

RECEIVED: 19 Aug 2017 REVISED: 4 Feb 2018 ACCEPTED: 21 May 2018

DAVID EISENBUD:

de@msri.org

Mathematical Sciences Research Institute, Berkeley, CA, United States

JSAG **8** (2018), 61–70
dx.doi.org/10.2140/jsag.2018.8.61

The Journal of Software for
Algebra and Geometry

A Macaulay2 package for computations with rational maps

GIOVANNI STAGLIANÒ

ABSTRACT: The Macaulay2 package `Cremona.m2` performs some computations on rational and birational maps between irreducible projective varieties. For instance, it provides methods to compute degrees and projective degrees of rational maps without any theoretical limitation, from which is derived a general method to compute the push-forward to projective space of Segre classes. Moreover, the computations can be done both deterministically and probabilistically. We give here a brief description of the methods and algorithms implemented.

INTRODUCTION. In this note we describe the computational package `Cremona.m2`, included with [Macaulay2] since version 1.9. A first rudimentary version of this package has been already used in an essential way in [Staglianò 2016] (it was originally named `bir.m2`), and recent applications can be found in [Staglianò 2018; Russo and Staglianò 2017]. Here we describe version 4.2.2 of the package.

`Cremona.m2` performs computations on rational and birational maps between absolutely irreducible projective varieties over a field $\mathbb{K}$. Among other things, it provides general methods to compute projective degrees of rational maps, from which, as is well known (see Proposition 1.2), one can interpret them as methods to compute the push-forward to projective space of Segre classes. The algorithms are naively derived from the mathematical definitions, with the advantages of being obvious, quite general and easily implemented. Moreover, all the methods (where this may make sense) are available both in a probabilistic version and in a deterministic version, and one can switch from one to the other with a boolean option named `MathMode`.

In Section 1, we will describe the main methods provided by the package and the algorithms implemented. Most of these have already been described in [Staglianò 2016, Section 2], but here we will consider a more general setting. For instance, Algorithm 1.3 for computing homogeneous components of kernels of homogeneous ring maps was presented in [Staglianò 2016, Algorithm 2.5] requiring that

MSC2010: 14E05, 14Q15.
Keywords: rational map, birational map, projective degrees, Segre class.
Cremona.m2 version 4.2.2

the map was between polynomial rings. In Section 2, we will show how these methods work in some particular examples, concluding with an experimental comparison of the running times of one of these methods with the corresponding ones proposed in [Helmer 2016] and [Harris 2017] (see also [Jost 2015]). For further technical details we refer to the documentation of the package, which can be shown using the command `viewHelp Cremona`.

We mention that the package `RationalMaps.m2`, by K. Schwede, D. Smolkin, S. H. Hassanzadeh, and C. J. Bott, is another package included with Macaulay2 for working with rational maps. It mainly focuses on providing a general method for inverting birational maps, which in some cases turns out to be competitive with the corresponding method of `Cremona.m2`.

1. DESCRIPTION OF THE MAIN METHODS. Throughout, we shall use the following notation. Let $\mathbb{K}$ denote a field; in practice, it can be for instance $\mathbb{Q}$, a finite field, or a fraction field of a polynomial ring over these. Let $\phi : X \dashrightarrow Y$ be a rational map from a subvariety $X = V(I) \subseteq \mathbb{P}^n = \mathrm{Proj}(\mathbb{K}[x_0, \ldots, x_n])$ to a subvariety $Y = V(J) \subseteq \mathbb{P}^m = \mathrm{Proj}(\mathbb{K}[y_0, \ldots, y_m])$, which can be represented, although not uniquely, by a homogeneous ring map

$$\varphi : \mathbb{K}[y_0, \ldots, y_m]/J \to \mathbb{K}[x_0, \ldots, x_n]/I$$

of quotients of polynomial rings by homogeneous ideals. Sometimes we will denote by $F_0, \ldots, F_m \in \mathbb{K}[x_0, \ldots, x_n]$ homogeneous forms of the same degree such that $\overline{F_i} := F_i + I = \varphi(y_i)$, for $i = 0, \ldots, m$. The common degree of these elements will be denoted by δ.

From algebraic geometry to computational algebra. For each homogeneous ideal $\mathfrak{a} \subseteq \mathbb{K}[x_0, \ldots, x_n]/I$ (resp. $\mathfrak{b} \subseteq \mathbb{K}[y_0, \ldots, y_m]/J$), we have a closed subscheme $V(\mathfrak{a}) \subseteq X$ (resp. $V(\mathfrak{b}) \subseteq Y$), and the following basic formulae hold:[1]

$$\overline{\phi(V(\mathfrak{a}))} = V(\varphi^{-1}(\mathfrak{a})) \quad \text{and} \quad \overline{\phi^{-1}(V(\mathfrak{b}))} = V((\varphi(\mathfrak{b})) : (\varphi(y_0), \ldots, \varphi(y_m))^{\infty}). \quad (1\text{-}1)$$

In particular, the (closure of the) image of ϕ is defined by the kernel of φ. Several issues concerning rational maps lead naturally to an examination of the left-hand sides of (1-1), and the right-hand sides of (1-1) can be determined using Gröbner basis techniques, whenever $\mathfrak{a}$ and $\mathfrak{b}$ are explicitly given. Furthermore, Macaulay2 provides useful commands such as `preimage`, `kernel` and `saturate`, so that the required user skill level is quite low. The aim of the package `Cremona.m2` is to provide further tools.

[1] By abuse of notation, we consider ϕ as a morphism defined on the open set $X \setminus V(\overline{F_0}, \ldots, \overline{F_m})$.

Computing projective degrees. The projective degrees are the most basic invariants of a rational map. Many others can be derived from them, such as, for instance, the dimension and the degree of the base locus. For more details on the subject, see [Harris 1992, Example 19.4, p. 240].

Definition 1.1 (projective degrees, [Harris 1992]). (1) The projective degrees $d_0(\phi), d_1(\phi), \ldots, d_{\dim X}(\phi)$ of the map ϕ are defined as the components of the multidegree of the closure of the graph $\Gamma_\phi \subset \mathbb{P}^n \times \mathbb{P}^m$.

(2) Equivalently, the i-th projective degree $d_i(\phi)$ can be defined in terms of dimension and degree of the closure of $\phi^{-1}(L)$, where L is a general $(m-\dim X+i)$-dimensional linear subspace of $\mathbb{P}^m$; more precisely, $d_i(\phi) = \deg \overline{\phi^{-1}(L)}$ if $\dim \overline{\phi^{-1}(L)} = i$, and $d_i(\phi) = 0$ otherwise.

In common computer algebra systems such as Macaulay2, it is easy to translate Definition 1.1 into code. We now describe in more detail how this can be done. All of this is implemented in the method `projectiveDegrees`; see Example 2.2 for an example using it.

Deterministic approach. Taking into account Definition 1.1(1), a bihomogeneous ideal for Γ_ϕ in $\mathbb{K}[x_0, \ldots, x_n, y_0, \ldots, y_m]$ can be, for instance, obtained as

$$(I + (\{y_i\, F_j - y_j\, F_i,\ 0 \le i, j \le m\})) : (F_0, \ldots, F_m)^\infty. \tag{1-2}$$

Therefore its multidegree can be computed in Macaulay2 with `multidegree`, which implements an algorithm according to [Miller and Sturmfels 2005, p. 165].

Probabilistic approach. (See also [Staglianò 2016, Remark 2.4].) Taking into account Definition 1.1(2), if L is defined by an ideal I_L, the second formula of (1-1) tells us that $\overline{\phi^{-1}(L)}$ is defined by the saturation of the ideal $(\varphi(I_L))$ by $(\overline{F_0}, \ldots, \overline{F_m})$ in the ring $\mathbb{K}[x_0, \ldots, x_n]/I$. So replacing the word *general* with *random* in the definition, we get a probabilistic algorithm that computes all the projective degrees. Moreover, we can considerably speed up this algorithm by taking into account two remarks: firstly, the saturation

$$\varphi(I_L) : (\overline{F_0}, \ldots, \overline{F_m})^\infty$$

is the same as

$$\varphi(I_L) : (\lambda_0\, \overline{F_0} + \cdots + \lambda_m\, \overline{F_m})^\infty,$$

where $\lambda_0, \ldots, \lambda_m \in \mathbb{K}$ are general scalars; secondly, the i-th projective degree of ϕ coincides with the $(i-1)$-th projective degree of the restriction of ϕ to a general hyperplane section of X.

An alternative deterministic approach. Replacing the word *general* with *symbolic* in Definition 1.1(2) gives us a deterministic algorithm for computing projective degrees. For instance, in the case in which $\phi : \mathbb{P}^n \dashrightarrow \mathbb{P}^n$ is a dominant rational

map, extending $\mathbb{K}$ to the fractional field of a polynomial ring $\mathbb{K}[a_0, \ldots, a_n]$, we have that $d_0(\phi)$ is the degree of the fiber of ϕ at the *symbolic point* $[a_0, \ldots, a_n]$.

Some applications using projective degrees.

The degree of a rational map. The degree of the map $\phi : X \dashrightarrow Y$ is the number of isolated points in the inverse image of a general point of $\overline{\phi(X)}$ over the algebraic closure of $\mathbb{K}$. This is the same as the ratio of $d_0(\phi)$ and $\deg \overline{\phi(X)}$, and thus it can be explicitly computed. Let us note, however, that in several cases we do not need to compute the kernel of φ. For instance, if X is a projective space, we are able to pick an abundance of rational points of $\overline{\varphi(X)}$ and then we apply the second formula of (1-1). Another special case is when $d_0(\phi)$ is a prime number: here we have only to establish if the image of ϕ is a linear subspace (e.g., applying Algorithm 1.3 with $d = 1$). The method provided by `Cremona.m2` for this computation is named `degreeOfRationalMap`.[2]

Methods related to this are `isBirational` and `isDominant`, with obvious meaning. The latter does not compute the kernel of φ, but it uses an algorithm that looks for $d_r(\phi)$, where $r = \dim X - \dim Y$. More precisely, the algorithm is based on the following fact: let $Z \subset Y$ be a random 0-dimensional linear section of Y; if $\dim \overline{\phi^{-1}(Z)} = \dim X - \dim Y \geq 0$, then ϕ is certainly dominant, otherwise it is probably not dominant (see [Mumford 1988, Chapter I, § 8] or [Hartshorne 1977, Chapter II, Exercise 3.22]). When this last case occurs, it is generally easy to find a nonzero element in the kernel of φ, and so this method turns out to be very effective even in its deterministic version (see Example 2.1).

The Segre class. It is well known that one can deduce an algorithm computing the push-forward to projective space of Segre classes from an algorithm computing projective degrees of rational maps between projective varieties and vice versa. Indeed, with our notation, we have the following:

Proposition 1.2 ([Fulton 1984, Proposition 4.4]; see also [Dolgachev 2011, Section 2.3; Aluffi 2003, Section 3]). *Let $\mathfrak{B} \subset X$ be the subscheme defined by $\overline{F_0}, \ldots, \overline{F_m}$ and let $\nu : X \hookrightarrow \mathbb{P}^n$ be the inclusion. If H denotes the hyperplane class of $\mathbb{P}^n$ and $r = \dim X$, then the push-forward $\nu_*(s(\mathfrak{B}, X))$ of the Segre class of $\mathfrak{B}$ in X is*

$$\nu_*(s(\mathfrak{B}, X)) = \sum_{k=0}^{\dim \mathfrak{B}} ((-1)^{r-k-1} \sum_{i=0}^{r-k} (-1)^i \binom{r-k}{i} \delta^{r-k-i}\, d_{r-i}(\varphi))\, H^{n-k}. \quad (1\text{-}3)$$

The general method `SegreClass`, provided by `Cremona.m2` for computing the push-forward to projective spaces of Segre classes, does basically nothing more

[2] Notice that, in general, if the result of the probabilistic algorithm for `degreeOfRationalMap` is wrong, it can be either too small or too large. However, as a consequence of [Hartshorne 1977, Chapter III, Exercise 10.9], it should always provide a lower bound when the map is dominant between smooth varieties.

than apply (1-3); see Example 2.3 for an example using this method. Furthermore, applying one of the main results in [Aluffi 2003], a method is derived to compute the push-forward to projective space of the Chern–Schwartz–MacPherson class CSM(W) of the support of a projective scheme W; recall that the component of dimension 0 of CSM(W) is the topological Euler characteristic of the support of W.

Computing homogeneous components of kernels. To compute, using Macaulay2, the homogeneous component of degree d of the kernel of `phi` ($= \varphi$), one can perform the command `ideal image basis(d,kernel phi)`. This is equivalent to the command `kernel(phi,d)` provided by `Cremona.m2`, but the latter uses the following obvious algorithm.

Algorithm 1.3. `Input:` the ring map φ and an integer d.

`Output:` homogeneous component of degree d of the kernel of φ.

- Find vector space bases $G_0, \ldots, G_r$ of $(\mathbb{K}[y_0, \ldots, y_m]/J)_d$ and $H_0, \ldots, H_s$ of $I_{d\,\delta}$, where subscripts stand for homogeneous components.
- Take generic linear combinations $\mathbf{G} = \sum_{i=0}^{r} a_i\, G_i$ and $\mathbf{H} = \sum_{j=0}^{s} b_j\, H_j$, and find a basis of solutions for the homogeneous linear system obtained by requiring that the polynomial

$$\mathbf{G}(F_0, \ldots, F_m) - \mathbf{H} \in \mathbb{K}[a_0, \ldots, a_r, b_0, \ldots, b_s][x_0, \ldots, x_n]$$

  vanishes identically.
- For each vector $(\hat{a_0}, \ldots, \hat{a_r}, \hat{b_0}, \ldots, \hat{b_s}) \in \mathbb{K}^{r+s+2}$ obtained in the previous step, replace in $\mathbf{G}$ the coefficients $a_0, \ldots, a_r$ with $\hat{a_0}, \ldots, \hat{a_r}$; return all these elements.

For small values of d, applying Algorithm 1.3 may turn out to be much faster than computing a list of generators of the kernel of the map; see for instance Example 2.1 below.

Inverting birational maps. General algorithms for inverting birational maps are known. One of them is implemented in the package `Parametrization.m2` by J. Boehm, and the method `inverseMap` of `Cremona.m2` uses the same one for the general case as well. However, when the source X of the rational map ϕ is a projective space and a further technical condition is satisfied, then it uses the following powerful algorithm.

Algorithm 1.4 ([Russo and Simis 2001]; see also [Simis 2004]). `Input:` the ring map φ (assuming that ϕ is birational and further conditions are satisfied).

`Output:` a ring map representing the inverse map of ϕ.

- Find generators $\{(L_{0,j}, \ldots, L_{m,j})\}_{j=1,\ldots,q}$ for the module of linear syzygies of $(F_0, \ldots, F_m)$.
- Compute the Jacobian matrix Θ of the bihomogeneous forms
$$\left\{\sum_{i=0}^{m} y_i\, L_{i,j}\right\}_{j=1,\ldots,q}$$
with respect to the variables $x_0, \ldots, x_n$ and consider the map of graded free modules $(\Theta \bmod J) : (k[y_0, \ldots, y_m]/J)^{n+1} \to (k[y_0, \ldots, y_m]/J)^q$.
- Return the map defined by a generator $G = (G_0, \ldots, G_n)$ for the kernel of $(\Theta \bmod J)$.

Remark 1.5. One of the main features of the package `RationalMaps.m2`, by Schwede, Smolkin, Hassanzadeh, and Bott, is a method for inverting birational maps, which, in the case when Algorithm 1.4 does not apply, appears to be quite competitive with the method `inverseMap` of `Cremona.m2`.

Heuristic approach. The method `approximateInverseMap` provides a heuristic approach to compute the inverse of a birational map modulo a change of coordinate. The idea of the algorithm is to try to construct the base locus of the inverse by looking for the images of general linear sections. Consider, for simplicity, the case in which $\phi : \mathbb{P}^n \dashrightarrow \mathbb{P}^{n'}$ is a Cremona transformation. Then, by taking the images of $n+1$ general hyperplanes in $\mathbb{P}^n$, we form a linear system of hypersurfaces in $\mathbb{P}^{n'}$ of degree $d_1(\phi)$ which defines a rational map $\psi : \mathbb{P}^{n'} \dashrightarrow \mathbb{P}^n$ such that $\psi \circ \phi$ is a (linear) isomorphism; i.e., we find an *approximation* of ϕ^{-1}. Next, we can fix the *error of the approximation* by observing that we have $\phi^{-1} = (\psi \circ \phi)^{-1} \circ \psi$. It is surprising that this method turns to be effective in examples where other deterministic algorithms seem to run endlessly; see for instance Example 2.1 below.

2. EXAMPLES. In this section, we show how the methods described in Section 1 can be applied in some particular examples. We note that the package `Cremona.m2` provides the data type `RationalMap`, but here we will use the more familiar type `RingMap`. For brevity, we will omit irrelevant output lines. We start with an example reviewing the construction given in [Staglianò 2016] of a quadro-quadric Cremona transformation of $\mathbb{P}^{20}$.

Example 2.1. The code below constructs a ring map `psi` representing a rational map $\psi : \mathbb{P}^{16} \dashrightarrow \mathbb{P}^{20}$. Precisely, the algorithm for constructing ψ is as follows: take $E \subset \mathbb{P}^7$ to be a 3-dimensional edge variety of degree 7, namely, the residual intersection of $\mathbb{P}^1 \times \mathbb{P}^3 \subset \mathbb{P}^7$ with a general quadric in $\mathbb{P}^7$ containing one of the $\mathbb{P}^3$'s of the rulings of $\mathbb{P}^1 \times \mathbb{P}^3 \subset \mathbb{P}^7$; next, see $E \subset \mathbb{P}^7$ embedded in a hyperplane of $\mathbb{P}^8$ and take the birational map $\phi : \mathbb{P}^8 \dashrightarrow \mathbb{P}^{16}$ defined by the quadrics of $\mathbb{P}^8$ containing E; take $\psi : \mathbb{P}^{16} \dashrightarrow \mathbb{P}^{20}$ to be the map defined by the quadrics of $\mathbb{P}^{16}$

containing the image of ϕ. For the first part of this construction, we use the package `Cremona.m2` only to shorten the code.

```
Macaulay2, version 1.11
with packages: ConwayPolynomials, Elimination, IntegralClosure, InverseSystems,
               LLLBases, PrimaryDecomposition, ReesAlgebra, TangentCone
i1 : loadPackage "Cremona";
i2 : K = ZZ/70001;
i3 : P8 = K[t_0..t_8]; E = saturate(minors(2,genericMatrix(P8,4,2))+sum(
     (ideal(t_0..t_7)*ideal(t_0..t_3))_*,u->random(K)*u),ideal(t_0..t_3)) + t_8;
i5 : psi = toMap kernel(toMap(E,2),2);
```

Up to this point, the computation was standard. But now we want to determine the homogeneous ideal of $Z := \overline{\psi(\mathbb{P}^{16})} \subset \mathbb{P}^{20}$, which turns out to be generated by quadrics. Computing this using `kernel psi` seems an impossible task, but it is elementary using `kernel(psi,2)`. So we can consider ψ as a dominant rational map $\psi : \mathbb{P}^{16} \dashrightarrow Z \subset \mathbb{P}^{20}$.

```
i6 : time Z = kernel(psi,2);
     -- used 2.84998 seconds
i7 : psi = toMap(psi,Dominant=>Z);
```

The map ψ turns out to be not only dominant but birational.

```
i8 : time degreeOfRationalMap psi
     -- used 2.11216 seconds
o8 = 1
```

We now want to compute the inverse of ψ. This is a case where `inverseMap` can apply Algorithm 1.4, but the running time is several hours. We can perform this computation in seconds by using `approximateInverseMap`.

```
i9 : time psi' = approximateInverseMap(psi,CodimBsInv=>10,MathMode=>true);
     -- used 15.9724 seconds
```

A Cremona transformation ω of $\mathbb{P}^{20}$ is then obtained combining ψ^{-1} and Z as follows.

```
i10 : omega = toMap(lift(matrix psi',ring Z)|gens Z);
```

Even checking just the dominance of ω, by computing `kernel omega`, seems an impossible task, but it can be done quickly with `isDominant`.

```
i11 : time isDominant(omega,MathMode=>true)
     -- used 0.100369 seconds
o11 = true
```

We now check that our map is birational and find its inverse using Algorithm 1.4.

```
i12 : time isBirational omega
     -- used 0.0366468 seconds
o12 = true
i13 : time inverseMap omega;
     -- used 0.0717518 seconds
```

Example 2.2. Here, we use the probabilistic versions of some methods. Take M to be a generic 3×5 matrix of linear forms on $\mathbb{P}^6$, and, let $\phi : \mathbb{P}^6 \dashrightarrow \mathbb{G}(2, 4) \subset \mathbb{P}^9$

be the rational map defined by the 3×3 minors of M (its base locus is a smooth threefold scroll over a plane).

```
i14 : P6 = K[x_0..x_6]; M = matrix pack(5,for i from 1 to 15 list random(1,P6));
i16 : phi = toMap(minors(3,M),Dominant=>2);
```

We check that the map is birational and compute its inverse.

```
i17 : time isBirational phi
     -- used 0.217607 seconds
o17 = true
i18 : time psi = inverseMap phi;
     -- used 1.39511 seconds
```

Now we compute the multidegrees of ϕ and ϕ^{-1}.

```
i19 : time (projectiveDegrees phi, projectiveDegrees psi)
     -- used 1.37582 seconds
o19 = ({1, 3, 9, 17, 21, 15, 5}, {5, 15, 21, 17, 9, 3, 1})
```

We also compute the push-forward to $\mathbb{P}^6$ (resp. $\mathbb{P}^9$) of the Segre class of the base locus of ϕ (resp. ϕ^{-1}) in $\mathbb{P}^6$ (resp. in $\mathbb{G}(2,4)$). As usual, H denotes the hyperplane class.

```
i20 : time (SegreClass phi, SegreClass psi)
     -- used 1.43359 seconds
           6       5      4      3      9       8       7      6      5
o20 = (- 680H  + 228H  - 60H  + 10H , 728H  - 588H  + 276H  - 98H  + 24H )
```

Example 2.3. In this example, we use the deterministic version of the method `SegreClass`. We take $Y \subset \mathbb{P}^{11}$ to be the dual quartic hypersurface of

$$\mathbb{P}^1 \times Q^4 \subset \mathbb{P}^{11*},$$

where $Q^4 \subset \mathbb{P}^5$ is a smooth quadric hypersurface, and take $X \subset Y$ to be the singular locus of Y. We then compute the push-forward to the Chow ring of $\mathbb{P}^{11}$ of the Segre class both of X in Y and of X in $\mathbb{P}^{11}$ working over the Galois field GF(331^2).

```
i21 : P11 = GF(331^2)[x_0..x_11];
i22 : Y = ideal sum(first entries gens minors(2,genericMatrix(P11,6,2)),t->t^2);
i23 : X = sub(ideal jacobian Y,P11/Y);
i24 : time SegreClass(X, MathMode=>true) -- push-forward of s(X,Y)
     -- used 0.789986 seconds
           11          10         9        8        7      6        5      4      3
o24 = 507384H   - 137052H   + 35532H  - 9018H  + 2340H  - 658H  + 204H  - 64H  + 16H
i25 : time SegreClass(lift(X,P11), MathMode=>true)  -- push-forward of s(X,P^11)
     -- used 0.846234 seconds
           11          10         9        8        7      6        5      4     3
o25 = 313568H   - 101712H   + 30636H  - 8866H  + 2532H  - 720H  +  198H  - 48H  + 8H
```

Example 2.4. Here we experimentally measure the probability of obtaining an incorrect answer using the probabilistic version of the method `projectiveDegrees` with a simple example of a birational map $\phi : \mathbb{G}(1,3) \dashrightarrow \mathbb{P}^4$ defined over $\mathbb{K}$. We define a procedure which computes this probability as a function of the field $\mathbb{K}$.

field	$\mathbb{Q}$	$\mathbb{Z}/70001$	GF(3^8)	$\mathbb{Z}/101$	$\mathbb{Z}/31$
probability	0.0	0.0	0.002	0.074	0.253

Table 1. Incorrect outputs of a probabilistic method.

In Table 1, we report the results obtained by running the procedure with various fields.

```
i26 : p = (K) -> (
         x := local x; R := K[x_0..x_4];
         phi := inverseMap toMap(minors(2,matrix{{x_0..x_3},{x_1..x_4}}),Dominant=>2);
         m := projectiveDegrees(phi,MathMode=>true);
         0.1 * # select(1000,i -> projectiveDegrees phi != m));
```

Example 2.5. Lastly, we deal with an experimental comparison of the method `SegreClass` of `Cremona.m2` and the corresponding ones of other Macaulay2 packages. Precisely, we want to compare the method `SegreClass` against the corresponding methods of the packages `CharacteristicClasses.m2` version 2.0, by M. Helmer and C. Jost (see [Helmer 2016; Jost 2015]), which provides a probabilistic method; and `FMPIntersectionTheory.m2` version 0.1, by C. Harris [2017], which provides a deterministic method. Since the former puts restrictions on the ambient variety, we will only consider examples where the ambient is a projective space. We are unable to determine precisely which is the fastest among all the methods and which, in the probabilistic case, has highest probability of giving the correct answer. We just summarize in Table 2 the running times for some special examples. Below is the code from which we obtained the first row of the table.

```
i27 : loadPackage "CharacteristicClasses"; loadPackage "FMPIntersectionTheory";
i29 : X = last(P5=ZZ/16411[vars(0..5)],ideal(random(3,P5),random(3,P5),random(4,P5)));
i30 : (time Segre X,time SegreClass X,time segreClass X,time SegreClass(X,MathMode=>true));
     -- used 0.1511 seconds
     -- used 1.00936 seconds
     -- used 34.1471 seconds
     -- used 74.572 seconds
```

input	CC	Cremona (prob.)	FMPIntTh	Cremona (det.)
complete int. of type (3, 3, 4) in $\mathbb{P}^5$	0.15	1.01	34.15	74.57
rational normal surface $S(1,4) \subset \mathbb{P}^6_{\mathbb{Q}}$	1.41	0.74	5.32	0.06
Grassmannian $\mathbb{G}(1,4) \subset \mathbb{P}^9_{\mathbb{Q}}$	0.16	0.09	0.42	0.02
base locus of ϕ in Ex. 2.2	0.23	0.44	6.49	663.79
$X \subset \mathbb{P}^{11}$ in Ex. 2.3 over $\mathbb{F}_{331^2}$	65.76	83.66	–	0.85
$X \subset \mathbb{P}^{11}$ in Ex. 2.3 over $\mathbb{Z}/16411$	3.62	11.61	198.92	0.74

Table 2. Run-times to compute Segre classes in `CharacteristicClasses.m2`, `FMPIntersectionTheory.m2`, and `Cremona.m2` (all times given in seconds).

SUPPLEMENT. The online supplement contains version 4.2.2 of `Cremona.m2`.

REFERENCES.

[Aluffi 2003] P. Aluffi, "Computing characteristic classes of projective schemes", *J. Symbolic Comput.* **35**:1 (2003), 3–19. MR Zbl

[Dolgachev 2011] I. Dolgachev, "Lectures on Cremona transformations", 2011, available at http://www.math.lsa.umich.edu/∼idolga/cremonalect.pdf.

[Fulton 1984] W. Fulton, *Intersection theory*, Ergebnisse der Mathematik und ihrer Grenzgebiete (3) **2**, Springer, 1984. MR Zbl

[Harris 1992] J. Harris, *Algebraic geometry, a first course*, Graduate Texts in Mathematics **133**, Springer, 1992. MR Zbl

[Harris 2017] C. Harris, "Computing Segre classes in arbitrary projective varieties", *J. Symbolic Comput.* **82** (2017), 26–37. MR Zbl

[Hartshorne 1977] R. Hartshorne, *Algebraic geometry*, Graduate Texts in Mathematics **52**, Springer, 1977. MR Zbl

[Helmer 2016] M. Helmer, "Algorithms to compute the topological Euler characteristic, Chern–Schwartz–MacPherson class and Segre class of projective varieties", *J. Symbolic Comput.* **73** (2016), 120–138. MR Zbl

[Jost 2015] C. Jost, "Computing characteristic classes and the topological Euler characteristic of complex projective schemes", *J. Softw. Algebra Geom.* **7** (2015), 31–39. MR

[Macaulay2] D. R. Grayson and M. E. Stillman, "Macaulay2: a software system for research in algebraic geometry", available at http://www.math.uiuc.edu/Macaulay2.

[Miller and Sturmfels 2005] E. Miller and B. Sturmfels, *Combinatorial commutative algebra*, Graduate Texts in Mathematics **227**, Springer, 2005. MR Zbl

[Mumford 1988] D. Mumford, *The red book of varieties and schemes*, Lecture Notes in Mathematics **1358**, Springer, 1988. MR Zbl

[Russo and Simis 2001] F. Russo and A. Simis, "On birational maps and Jacobian matrices", *Compositio Math.* **126**:3 (2001), 335–358. MR Zbl

[Russo and Staglianò 2017] F. Russo and G. Staglianò, "Congruences of 5-secant conics and the rationality of some admissible cubic fourfolds", 2017. arXiv

[Simis 2004] A. Simis, "Cremona transformations and some related algebras", *J. Algebra* **280**:1 (2004), 162–179. MR Zbl

[Staglianò 2016] G. Staglianò, "Examples of special quadratic birational transformations into complete intersections of quadrics", *J. Symbolic Comput.* **74** (2016), 635–649. MR Zbl

[Staglianò 2018] G. Staglianò, "Special cubic Cremona transformations of $\mathbb{P}^6$ and $\mathbb{P}^7$", *Adv. Geom.* (online publication March 2018).

RECEIVED: 19 Jan 2017 REVISED: 24 May 2018 ACCEPTED: 11 Jun 2018

GIOVANNI STAGLIANÒ:

giovannistagliano@gmail.com

Dipartimento di Ingegneria Industriale e Scienze Matematiche, Università Politecnica delle Marche, Ancona, Italy

JSAG **8** (2018), 71–79
dx.doi.org/10.2140/jsag.2018.8.71

The Journal of Software for
Algebra and Geometry

ExteriorIdeals: a package for computing monomial ideals in an exterior algebra

LUCA AMATA AND MARILENA CRUPI

ABSTRACT: Let K be a field, V a K-vector space with basis $e_1, \dots, e_n$, and E the exterior algebra of V. We introduce a *Macaulay2* package that allows one to deal with classes of monomial ideals in E. More precisely, we implement in *Macaulay2* some algorithms in order to easily compute stable, strongly stable and lexsegment ideals in E. Moreover, an algorithm to check whether an $(n+1)$-tuple $(1, h_1, \dots, h_n)$ $(h_1 \leq n = \dim_K V)$ of nonnegative integers is the Hilbert function of a graded K-algebra of the form E/I, with I a graded ideal of E, is given. In particular, if $H_{E/I}$ is the Hilbert function of a graded K-algebra E/I, the package is able to construct the unique lexsegment ideal I^{lex} such that $H_{E/I} = H_{E/I^{\text{lex}}}$.

1. INTRODUCTION. Monomial ideals are a bridge between algebra and combinatorial algebra. It is well known that, even if such ideals are, in some sense, among the simplest structures in commutative algebra, they are the main objects of combinatorial commutative algebra. Many authors have focused their attention on classes of monomials ideals in an exterior algebra [Aramova et al. 1997; 2000; Crupi and Utano 1999; 2007; Crupi and Ferró 2015; Eisenbud et al. 2003; Crupi 2015; Gasharov 1997; Murai 2011; Shakin 2004; 2005] and on the behavior of certain invariants, such as for instance, the Hilbert function.

In this paper, we introduce `ExteriorIdeals.m2` — a new package written for [Macaulay2] for manipulating special classes of monomial ideals in an exterior algebra of a finite-dimensional vector space over a field. More precisely, the package provides functions to check whether a monomial ideal is stable, strongly stable, or lexsegment, and, respectively, to compute the smallest stable, strongly stable, or lexsegment ideal containing a given monomial ideal. Moreover, given an exterior algebra, the package allows the computation of all the Hilbert sequences of quotients of the exterior algebra. Some utility functions are necessary to simplify and optimize the implementation of the main algorithms, such as the Macaulay expansion, the initial degree of a graded ideal, the support of a monomial and the shadow of a set of monomials. Most of the algorithms must work in an exterior algebra

MSC2010: 13A02, 15A75, 68W30.
Keywords: exterior algebra, monomial ideals, Hilbert functions, algorithms.
ExteriorIdeals.m2 version 1.0

endowed with the lexicographic order, so that such an ordering is forced within routines. Nevertheless, the ideals obtained by our algorithms are made compatible with the native exterior algebra to allow further computations.

2. MATHEMATICAL BACKGROUND. Let K be a field. We denote by

$$E = K\langle e_1, \dots, e_n\rangle$$

the exterior algebra of a K-vector space V with basis $e_1, \dots, e_n$. For any subset $\sigma = \{i_1, \dots, i_d\}$ of $\{1, \dots, n\}$, with $i_1 < i_2 < \cdots < i_d$, we write $e_\sigma = e_{i_1} \wedge \cdots \wedge e_{i_d}$, and call e_σ a monomial of degree d. We set $e_\sigma = 1$, if $\sigma = \varnothing$. The set of monomials in E forms a K-basis of E of cardinality 2^n.

In order to simplify the notation, we write $fg = f \wedge g$ for any two elements f and g in E. An element $f \in E$ is called *homogeneous* of degree j if $f \in E_j$, where $E_j = \bigwedge^j V$. An ideal I is called *graded* if I is generated by homogeneous elements. If I is graded, then $I = \oplus_{j\geq 0} I_j$, where I_j is the K-vector space of all homogeneous elements $f \in I$ of degree j. We denote by $\operatorname{indeg}(I)$ the *initial degree* of I, i.e., the least degree of a homogeneous generator of I.

If I is a graded ideal in E, then the function $H_I : \mathbb{N} \to \mathbb{N}$ given by $H_I(d) = \dim_K I_d$ $(i \geq 0)$ is called the Hilbert function of I.

Now let $e_\sigma = e_{i_1} \cdots e_{i_d} \neq 1$ be a monomial in E. We define

$$\operatorname{supp}(e_\sigma) = \sigma = \{j : e_j \text{ divides } e_\sigma\}, \quad \mathrm{m}(e_\sigma) = \max\{i : i \in \operatorname{supp}(e_\sigma)\}.$$

Moreover, we set $\mathrm{m}(e_\sigma) = 0$ if $e_\sigma = 1$.

If M is a set of monomials of degree $d < n$ of E, the set of monomials of degree $d + 1$,

$$\operatorname{Shad}(M) = \{(-1)^{\alpha(\sigma, j)} e_j e_\sigma : e_\sigma \in M,\ j \notin \operatorname{supp}(e_\sigma),\quad j = 1, \dots, n\},$$

where $\alpha(\sigma, j) = |\{r \in \sigma : r < j\}|$, is called the *shadow* of M and is denoted by $\operatorname{Shad}(M)$ [Crupi and Ferró 2015, Definition 2.4].

Definition 2.1. Let I be a monomial ideal of E. I is called *stable* if for each monomial $e_\sigma \in I$ and each $j < \mathrm{m}(e_\sigma)$ one has $e_j e_{\sigma\setminus\{\mathrm{m}(e_\sigma)\}} \in I$. I is called *strongly stable* if for each monomial $e_\sigma \in I$ and each $j \in \sigma$ one has $e_i e_{\sigma\setminus\{j\}} \in I$, for all $i < j$.

If I is a monomial ideal of E, we denote by $G(I)$ the unique minimal set of monomial generators of I.

Remark 2.2. One can observe that the defining property of a strongly stable ideal needs to be checked only for the set of monomial generators of a monomial ideal. Indeed, let I be a monomial ideal and suppose that for all $e_\sigma \in G(I)$, and for all integers $1 \leq i < j \leq n$ such that $j \in \sigma$, one has $e_i e_{\sigma\setminus\{j\}} \in I$. Then I is strongly stable.

Let $e_\tau \in I$ be a monomial and $1 \le i < j \le n$ be integers such that $j \in \tau$. There exist $e_\sigma \in G(I)$ and a monomial $e_\mu \in E$ such that $e_\tau = e_\sigma e_\mu$ in E.

We distinguish two cases: $j \in \sigma,\ j \in \mu$. If $j \in \sigma$, then $e_i e_{\sigma\setminus\{j\}} \in I$ by assumption, and so $e_i e_{\tau\setminus\{j\}} = e_i e_{\sigma\setminus\{j\}} e_\mu \in I$.

If $j \in \mu$, then $e_i e_{\tau\setminus\{j\}} = e_i e_\sigma e_{\mu\setminus\{j\}} \in I$.

Another class of monomial ideals which plays a relevant role in combinatorial commutative algebra is the class of *lexsegment ideals*. The lexsegment ideals provide an upper bound for the graded Betti numbers of graded ideals with given Hilbert function [Aramova et al. 1997, Theorem 4.4].

Let $\mathrm{Mon}_d(E)$ be the set of all monomials of degree $d \ge 1$ in E. Denote by $>_{\mathrm{lex}}$ the *lexicographic order* on $\mathrm{Mon}_d(E)$, i.e., if $e_\sigma = e_{i_1} e_{i_2} \cdots e_{i_d}$ and $e_\tau = e_{j_1} e_{j_2} \cdots e_{j_d}$ are monomials belonging to $\mathrm{Mon}_d(E)$, with $1 \le i_1 < i_2 < \cdots < i_d \le n$ and $1 \le j_1 < j_2 < \cdots < j_d \le n$, then $e_\sigma >_{\mathrm{lex}} e_\tau$ if $i_1 = j_1, \ldots, i_{s-1} = j_{s-1}$ and $i_s < j_s$ for some $1 \le s \le d$.

Definition 2.3. A nonempty subset M of $\mathrm{Mon}_d(E)$ is called a lexsegment of degree d if for all $v \in M$ and all $u \in \mathrm{Mon}_d(E)$ such that $u >_{\mathrm{lex}} v$, we have that $u \in M$.

Definition 2.4. A monomial ideal I of E is called a lexsegment ideal (lex ideal, for short) if for all monomials $u \in I$ and all monomials $v \in E$ with $\deg u = \deg v$ and $v >_{\mathrm{lex}} u$, we have $v \in I$.

Equivalently, a monomial ideal I in E is called a *lex ideal* if $\mathrm{Mon}_d(I)$ is a lexsegment for all d; $\mathrm{Mon}_d(I)$ is the set of all monomials of degree d in I.

Remark 2.5. Every lex ideal of E is obviously a strongly stable ideal, and consequently a stable ideal.

Now let a and i be two positive integers. Then a has the unique i-th Macaulay expansion [Herzog and Hibi 2011, Lemma 6.3.4]

$$a = \binom{a_i}{i} + \binom{a_{i-1}}{i-1} + \cdots + \binom{a_j}{j}$$

with $a_i > a_{i-1} > \cdots a_j \ge j \ge 1$. We define

$$a^{(i)} = \binom{a_i}{i+1} + \binom{a_{i-1}}{i} + \cdots + \binom{a_j}{j+1}.$$

We also set $0^{(i)} = 0$ for all $i \ge 1$.

The next theorem describes the possible Hilbert functions of graded K-algebras of the form E/I, with I a graded ideal in E. It is the precise analogue to Macaulay's theorem [Bruns and Herzog 1993; Eisenbud 1995] which describes the possible Hilbert functions of standard graded K-algebras.

Theorem 2.6 [Aramova et al. 1997, Theorem 4.1]. *Let* $(h_1, \ldots, h_n)$ *be a sequence of nonnegative integers. Then the following conditions are equivalent*:

(a) $1 + \sum_{i=1}^{n} h_i t^i$ *is the Hilbert series of a graded* K*-algebra* E/I.

(b) $0 < h_{i+1} \leq h_i^{(i)},\ 0 < i \leq n-1$.

Theorem 2.6 is known as the *Kruskal–Katona theorem*. Its proof points out that if I is a graded ideal of E, then there exists a unique lex ideal of E, usually denoted by I^{lex}, such that $H_{E/I} = H_{E/I^{\text{lex}}}$.

More precisely, if $(1, h_1, \ldots, h_n)$ is a sequence of nonnegative integers such that

(i) $h_1 \leq n$,

(ii) $0 < h_{i+1} \leq h_i^{(i)},\ 0 < i \leq n-1$,

then there exists a unique lex ideal J ($\text{indeg} J \geq 1$) of an exterior algebra E with n generators over a field K such that $H_{E/J}(d) = h_d$ $(d = 0, \ldots, n)$.

If $1 + \sum_{i=1}^{n} h_i t^i$ is the Hilbert series of a graded K-algebra E/I, then the sequence $(1, h_1, \ldots, h_n)$ is called the *Hilbert sequence* of E/I.

From the Kruskal–Katona theorem, one can deduce that a sequence of nonnegative integers $(h_0, h_1, \ldots, h_n)$ is the Hilbert sequence of a graded K-algebra E/I, with $I \subsetneq E$ a graded ideal of initial degree ≥ 1, if $h_0 = 1$, and (i) and (ii) hold.

From now on, when we speak about Hilbert sequences we refer to Hilbert sequences of quotients of an exterior algebra.

3. EXAMPLES. In this section, we collect some examples in order to describe the algorithms. Our implementation works in any characteristic.

Example 3.1. Given a monomial ideal I in an exterior algebra E, we illustrate how some functions from our package allow one to check whether I is stable, strongly stable, or lex, and to produce stable or strongly stable ideals containing I. The core of the algorithms is based on the fact that the minimal monomial generators of a stable or strongly stable ideal must satisfy the criterion in Definition 2.1 (see Remark 2.2) and on the fact that the shadow of a lexsegment of monomials is again a lexsegment [Herzog and Hibi 2011].

```
Macaulay2, version 1.10
with packages: ConwayPolynomials, Elimination, IntegralClosure,
InverseSystems, LLLBases, PrimaryDecomposition, ReesAlgebra,
TangentCone

i1 : loadPackage "ExteriorIdeals"
i2 : E=QQ[e_1..e_5,SkewCommutative=>true]
i3 : I=ideal {e_2*e_3,e_3*e_4*e_5}
o3 = ideal (e e , e e e )
             2 3   3 4 5
```

```
o3 : Ideal of E
i4 : isStableIdeal I
o4 = false
```

The ideal I is not stable. Indeed, the monomial e_1e_2 is not in I even though e_2e_3 is. Hence, by the function `stableIdeal(ideal)`, we compute the smallest stable ideal (Is) containing I:

```
i5 : Is=stableIdeal I
o5 = ideal (e e , e e e , e e , e e e )
            1 2   1 3 4   2 3   3 4 5
o5 : Ideal of E
i6 : isStableIdeal Is
o6 = true
i7 : isStronglyStableIdeal Is
o7 = false
```

The ideal Is is stable but not strongly stable in E. Note that the monomial e_1e_3 is not in Is even though e_2e_3 is.

Using the function `stronglyStableIdeal(ideal)`, we compute the smallest strongly stable ideal (Iss) containing Is, and consequently I:

```
i8 : Iss=stronglyStableIdeal Is
o8 = ideal (e e , e e , e e e , e e , e e e , e e e )
            1 2   1 3   1 4 5   2 3   2 4 5   3 4 5
o8 : Ideal of E
i9 : isStronglyStableIdeal Iss
o9 = true
i10 : Iss2=stronglyStableIdeal I
o10 = ideal (e e , e e , e e e , e e , e e e , e e e )
             1 2   1 3   1 4 5   2 3   2 4 5   3 4 5
o10 : Ideal of E
i11 : Iss==Iss2
o11 = true
```

The ideal Iss is not a lex ideal in E. Indeed, the monomial e_1e_4 does not belong to Iss, but $e_1e_4 >_{\text{lex}} e_2e_3$. One can verify this by the function `isLexIdeal(ideal)`:

```
i12 : isLexIdeal Iss
o12 = false
```

Example 3.2. Letting E be an exterior algebra with n generators over a field K and $h = (h_0, h_1, \ldots, h_n)$ be a sequence of nonnegative integers, we describe how one can verify if h is a Hilbert sequence.

The key tools in our algorithm are the functions `isHilbertSequence(list, exterior algebra)` and `lexIdeal(list,exterior algebra)`. The first function verifies if a list of nonnegative integers of length $n+1$ is a Hilbert function; the

second one returns a lex ideal of E if and only if the list is a Hilbert sequence. In more detail, if $(h_0, h_1, \ldots, h_n)$ is a Hilbert sequence, the lex ideal of E produced by the function `lexIdeal({`$h_0, \ldots, h_n$`}, E)` is the unique lex ideal I of E with $H_{E/I}(d) = h_d$ $(d = 0, \ldots, n)$. The procedure for the computation of the required lex ideal is based on the constructive proof of Theorem 2.6 (see [Aramova et al. 1997, Theorem 4.1, (b) $\Rightarrow$ (a)]).

We start with some examples of sequences which are not Hilbert sequences. The property is verified by using either `isHilbertSequence(list,exterior algebra)` or `lexIdeal(list,exterior algebra)`:

```
Macaulay2, version 1.10
with packages: ConwayPolynomials, Elimination, IntegralClosure,
InverseSystems, LLLBases, PrimaryDecomposition, ReesAlgebra,
TangentCone

i1 : loadPackage "ExteriorIdeals"
i2 : E=QQ[e_1..e_5,SkewCommutative=>true]
i3 : isHilbertSequence({2,4,3,0,0,0},E)
o3 = false
i4 : isHilbertSequence({0,4,3,0,0,0},E)
o4 : false
i5 : lexIdeal({1,6,3,0,0,0,0},E)
stdio:24:1:(3): error: expected a Hilbert sequence
i6 : lexIdeal({1,5,10,10,5,1,0},E)
stdio:26:1:(3): error: expected a Hilbert sequence
```

Moreover, the next statements provide some examples of the lex ideal produced by a Hilbert sequence. The length of the sequence can be at most $n + 1$; if the length is less than $n + 1$, then the sequence will be completed by adding zeros on the right.

```
i6 : lexIdeal({1,4,3,0,0,0},E)
o6 = ideal (e , e e , e e , e e , e e e )
             1   2 3   2 4   2 5   3 4 5
o6 : Ideal of E
i7 : lexIdeal({1,4,4},E)
o7 = ideal (e , e e , e e , e e e )
             1   2 3   2 4   3 4 5
o7 : Ideal of E
i8 : lexIdeal({1,5,7,4,0,0},E)
o9 = ideal (e e , e e , e e , e e e e )
             1 2   1 3   1 4   2 3 4 5
o9 : Ideal of E
```

The function `lexIdeal(list,exterior algebra)`, defined above, also plays a relevant role in the next algorithm.

Example 3.3. Given an exterior algebra E and a graded ideal I in E, we illustrate how to obtain the unique lex ideal I^{lex} with the same Hilbert function as I. In more detail, we describe two different methods for computing such a lex ideal.

```
Macaulay2, version 1.10
with packages: ConwayPolynomials, Elimination, IntegralClosure,
InverseSystems, LLLBases, PrimaryDecomposition, ReesAlgebra,
TangentCone

i1 : loadPackage "ExteriorIdeals";
i2 : E=QQ[e_1..e_5,SkewCommutative=>true]
i3 : I=ideal {e_1*e_2*e_3+e_3*e_4*e_5,e_1*e_3+e_4*e_5,e_2*e_3*e_4}
o3 = ideal (e e e  + e e e , e e  + e e , e e e )
             1 2 3    3 4 5   1 3    4 5   2 3 4
o3 : Ideal of E
i4 : hilbSeq=hilbertSequence(I)
o4 = {1, 5, 9, 3, 0, 0}
o4 : List
```

A first way for computing the lex ideal we are looking for is to use the function `lexIdeal(list,exterior algebra)`:

```
i5 : Ilex1=lexIdeal(hilbSeq,E)
o5 = ideal (e e , e e e , e e e , e e e , e e e)
             1 2   1 3 4   1 3 5   1 4 5   2 3 4
o5 : Ideal of E
i6 : isLexIdeal Ilex1
o6 = true
i7 : hilbertSequence(Ilex1)
o7 = {1, 5, 9, 3, 0, 0}
o7 : List
```

and a second one is *via* the new function `lexIdeal(ideal)`, which returns directly the required lex ideal:

```
i8 : Ilex2=lexIdeal(I)
o8 = ideal (e e , e e e , e e e , e e e , e e e)
             1 2   1 3 4   1 3 5   1 4 5   2 3 4
o8 : Ideal of E
i9 : hilbertSequence(Ilex2)
o9 = {1, 5, 9, 3, 0, 0}
o9 : List
```

Finally, our last example is related to the algorithm for the computation of Hilbert sequences.

Example 3.4. Given an exterior algebra E, we illustrate how to get all the Hilbert sequences of quotients of E.

```
Macaulay2, version 1.10
with packages: ConwayPolynomials, Elimination, IntegralClosure,
InverseSystems, LLLBases, PrimaryDecomposition, ReesAlgebra,
TangentCone
i1 : loadPackage "ExteriorIdeals";
i2 : E=QQ[e_1..e_4,SkewCommutative=>true]
i3 : hilbSeqs=allHilbertSequences(E)
o3 = {{1, 4, 6, 4, 1}, {1, 4, 6, 4, 0}, {1, 4, 6, 3, 0}, {1, 4, 6,2, 0},
     ------------------------------------------------------------------------
     {1, 4, 6, 1, 0}, {1, 4, 6,0, 0}, {1, 4, 5, 2, 0}, {1, 4, 5, 1, 0},
     ------------------------------------------------------------------------
     {1, 4, 5, 0, 0}, {1, 4, 4, 1, 0}, {1, 4, 4, 0, 0}, {1,4, 3, 1, 0},
     ------------------------------------------------------------------------
     {1, 4, 3, 0, 0}, {1, 4, 2, 0, 0}, {1, 4, 1, 0, 0}, {1, 4, 0, 0, 0},
     ------------------------------------------------------------------------
     {1, 3, 3, 1,0}, {1, 3, 3, 0, 0}, {1, 3, 2, 0, 0}, {1, 3, 1, 0, 0},
     ------------------------------------------------------------------------
     {1, 3, 0, 0, 0}, {1, 2, 1, 0, 0}, {1, 2,0, 0, 0}, {1, 1, 0, 0, 0},
     ------------------------------------------------------------------------
     {1, 0, 0, 0, 0}}
o3 : List
i4 : transpose matrix hilbSeqs
o4 = | 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 |
     | 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 3 3 3 3 3 2 2 1 0 |
     | 6 6 6 6 6 6 5 5 5 4 4 3 3 2 1 0 3 3 2 1 0 1 0 0 0 |
     | 4 4 3 2 1 0 2 1 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 |
     | 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 |
              5        25
o4 : Matrix ZZ  <--- ZZ
```

Note that the method `allHilbertSequences` returns an object of type `List`; for a more compact view it could be displayed as a matrix.

4. CONCLUSIONS AND PERSPECTIVES. The algorithms described in the examples above are part of a *Macaulay2* package `ExteriorIdeals.m2`, which has been tested with *Macaulay2* version 1.10. We are confident that this package may prove useful for further applications. Indeed, to the best of our knowledge, it seems that no packages for manipulating monomial ideals in an exterior algebra have been implemented, though functions for computing monomial ideals in a polynomial ring are available in many computer algebra systems (for instance, [CoCoA], [Macaulay2] and [Singular]).

We believe it would be nice to implement such a package for monomial modules over an exterior algebra. This task is currently under investigation by the authors.

ACKNOWLEDGEMENTS. The authors thank the anonymous referees for their useful comments that improved the quality of the paper.

SUPPLEMENT. The online supplement contains version 1.0 of `ExteriorIdeals.m2`.

REFERENCES.

[Aramova et al. 1997] A. Aramova, J. Herzog, and T. Hibi, "Gotzmann theorems for exterior algebras and combinatorics", *J. Algebra* **191**:1 (1997), 174–211. MR Zbl

[Aramova et al. 2000] A. Aramova, L. L. Avramov, and J. Herzog, "Resolutions of monomial ideals and cohomology over exterior algebras", *Trans. Amer. Math. Soc.* **352**:2 (2000), 579–594. MR Zbl

[Bruns and Herzog 1993] W. Bruns and J. Herzog, *Cohen–Macaulay rings*, Cambridge Studies in Advanced Mathematics **39**, Cambridge University Press, 1993. MR Zbl

[CoCoA] J. Abbott, A. M. Bigatti, and L. Robbiano, "CoCoA: a system for doing Computations in Commutative Algebra", available at http://cocoa.dima.unige.it.

[Crupi 2015] M. Crupi, "Algebraic invariants of graded ideals with a given Hilbert function in an exterior algebra", *Bull. Math. Soc. Sci. Math. Roumanie* (*N.S.*) **58(106)**:4 (2015), 393–403. MR Zbl

[Crupi and Ferró 2015] M. Crupi and C. Ferró, "Bounding Betti numbers of monomial ideals in the exterior algebra", *Pure Appl. Math. Q.* **11**:2 (2015), 267–281. MR Zbl

[Crupi and Utano 1999] M. Crupi and R. Utano, "Upper bounds for the Betti numbers of graded ideals of a given length in the exterior algebra", *Comm. Algebra* **27**:9 (1999), 4607–4631. MR Zbl

[Crupi and Utano 2007] M. Crupi and R. Utano, "Classes of graded ideals with given data in the exterior algebra", *Comm. Algebra* **35**:8 (2007), 2386–2408. MR Zbl

[Eisenbud 1995] D. Eisenbud, *Commutative algebra with a view toward algebraic geometry*, Graduate Texts in Mathematics **150**, Springer, 1995. MR Zbl

[Eisenbud et al. 2003] D. Eisenbud, S. Popescu, and S. Yuzvinsky, "Hyperplane arrangement cohomology and monomials in the exterior algebra", *Trans. Amer. Math. Soc.* **355**:11 (2003), 4365–4383. MR Zbl

[Gasharov 1997] V. Gasharov, "Extremal properties of Hilbert functions", *Illinois J. Math.* **41**:4 (1997), 612–629. MR Zbl

[Herzog and Hibi 2011] J. Herzog and T. Hibi, *Monomial ideals*, Graduate Texts in Mathematics **260**, Springer, 2011. MR Zbl

[Macaulay2] D. R. Grayson and M. E. Stillman, "Macaulay2: a software system for research in algebraic geometry", available at http://www.math.uiuc.edu/Macaulay2.

[Murai 2011] S. Murai, "Free resolutions of lex-ideals over a Koszul toric ring", *Trans. Amer. Math. Soc.* **363**:2 (2011), 857–885. MR Zbl

[Shakin 2004] D. A. Shakin, "Hilbert functions and Betti numbers of homogeneous ideals in an exterior algebra", *Uspekhi Mat. Nauk* **59**:5(359) (2004), 165–166. MR Zbl

[Shakin 2005] D. A. Shakin, "Piecewise-lexsegment ideals in exterior algebras", *Mat. Sb.* **196**:2 (2005), 287–307. MR Zbl

[Singular] W. Decker, G.-M. Greuel, G. Pfister, and H. Schönemann, "SINGULAR 4-1-1 — A computer algebra system for polynomial computations", available at http://www.singular.uni-kl.de.

RECEIVED: 28 Aug 2017 REVISED: 15 May 2018 ACCEPTED: 24 Jun 2018

LUCA AMATA:
lamata@unime.it
Department of Mathematics and Computer Sciences, Physics and Geological Sciences,
University of Messina, Messina, Italy

MARILENA CRUPI:
mcrupi@unime.it
Department of Mathematics and Computer Sciences, Physics and Geological Sciences,
University of Messina, Messina, Italy

JSAG **8** (2018), 81–86
dx.doi.org/10.2140/jsag.2018.8.81

The Journal of Software for
Algebra and Geometry

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

DAVID SWINARSKI

ABSTRACT: We introduce the packages `LieTypes.m2` and `ConformalBlocks.m2` for Macaulay2. `LieTypes.m2` contains basic types for working with Lie algebras and Lie algebra modules. `ConformalBlocks.m2` computes ranks and first Chern classes of vector bundles of conformal blocks on $\overline{M}_{0,n}$.

1. INTRODUCTION. The moduli stacks $\overline{\mathcal{M}}_{g,n}$ of Deligne–Mumford stable n-pointed curves of genus g are central objects of study in algebraic geometry and mathematical physics. The WZW model of conformal field theory can be interpreted as defining vector bundles on $\overline{\mathcal{M}}_{g,n}$ whose fibers are the so-called vector spaces of conformal blocks. These vector bundles were first constructed by Tsuchiya, Ueno [2008], and Yamada; their ranks are computed by the famous Verlinde formula.

We omit the lengthy full definition of conformal blocks (see the references [Beauville 1996] and [Ueno 2008]) and instead merely describe the input required to specify a conformal block. Let $\mathfrak{g}$ be a simple Lie algebra, and let ℓ be a positive integer called the *level*. Choose a set of simple roots for the root system associated to $\mathfrak{g}$, and let θ be the highest root. Let $(-,-)$ denote the Killing form, normalized so that $(\theta,\theta)=2$.

Proposition 1.1. *Let g and n be nonnegative integers satisfying $3g-3+n\geq 0$. Let ℓ be a positive integer. Let $\vec{\lambda}=(\lambda_1,\ldots,\lambda_n)$ be an n-tuple of weights with $(\lambda_i,\theta)\leq\ell$ for each $i=1,\ldots,n$. For each such triple $(\mathfrak{g},\ell,\vec{\lambda})$, we may construct a vector bundle $\mathbb{V}(\mathfrak{g},\ell,\vec{\lambda})$ on $\overline{\mathcal{M}}_{g,n}$, called the vector bundle of conformal blocks.*

In 2008, Fakhruddin gave formulas for the Chern classes of these vector bundles [Fakhruddin 2012]. We will refer to the first Chern class of a conformal block bundle as a *conformal block divisor*. The package `ConformalBlocks.m2` implements some of Fakhruddin's main formulas in the genus 0 case.

Several quantities from representation theory appear in Fakhruddin's formulas, and the earliest version of `ConformalBlocks.m2` contained several functions for

MSC2010: primary 14D21; secondary 14D22, 81T40.
Keywords: conformal blocks, fusion product, moduli of curves.
ConformalBlocks.m2 version 2.4
LieTypes.m2 version 0.5

representation theory calculations. At the suggestion of Grayson and Stillman, these were moved into a separate package, `LieTypes.m2`.

2. THE LIETYPES.M2 PACKAGE. The `LieTypes.m2` package defines two new classes, `LieAlgebra` and `LieAlgebraModule`; objects of these classes are hash tables. Currently, only simple Lie algebras over $\mathbb{C}$ are implemented. (Volunteers who would like to extend the functionality of this package are invited to contact the author.) Simple Lie algebras over $\mathbb{C}$ are specified by their rank and root system type. Irreducible Lie algebra modules are specified by their underlying Lie algebra and highest weight, and a general Lie algebra module is specified by the multiplicities of the irreducible submodules it contains.

The `LieTypes.m2` package contains several functions implementing basic Lie algebra data, such as the Cartan matrix. The documentation within the package contains references for formulas and/or sources of reference data for each of these functions. This package uses Macaulay2's combinatorial and linear algebra functions.

2.1. ***Tensor coefficients and fusion coefficients.*** One notable feature of the package `LieTypes.m2` is that it computes tensor product decompositions and fusion product decompositions for all irreducible root system types.

Let V_λ denote the irreducible $\mathfrak{g}$-module with highest weight λ. Define the tensor product coefficients $N_{\lambda,\mu}^{\nu}$ by

$$V_\lambda \otimes V_\mu = \bigoplus V_\nu^{\oplus N_{\lambda,\mu}^{\nu}}.$$

The `LieTypes.m2` package uses the Racah–Speiser algorithm for computing tensor product coefficients [Di Francesco et al. 1997, 13.5.2].

In type A (that is, $\mathfrak{g} = \mathfrak{sl}_k$), the tensor product coefficients are the Littlewood–Richardson coefficients. These coefficients have been previously implemented in other Macaulay2 packages (e.g., `SchurRings.m2`).

The fusion product $\otimes_\ell$ is a product for integrable level ℓ modules over an affine Lie algebra $\hat{g}$. The fusion coefficients $N_{\lambda,\mu}^{(\ell)\nu}$ are defined by the decomposition of the fusion product, and can be computed using the Kac–Walton algorithm (see [Di Francesco et al. 1997, § 16.2.2]). The Kac–Walton algorithm is closely related to the Racah–Speiser algorithm for tensor products, and it is defined entirely using the combinatorics of the root system of the underlying finite-dimensional Lie algebra. Therefore, we can abuse notation and use the Kac–Walton algorithm to define a product $\otimes_\ell$ on Lie algebra modules as well as affine Lie algebra modules.

Fusion coefficients have previously been implemented in KAC and Magma; but, to the author's knowledge, the implementation in `LieTypes.m2` in Macaulay2 is the first free, open-source implementation of fusion coefficients.

As an example, let $\mathfrak{g} = \mathfrak{sl}_3$. Let ω_1 and ω_2 be the fundamental dominant weights, and $\lambda = 2\omega_1 + \omega_2 = (2, 1)$, $\mu = \omega_1 + 2\omega_2 = (1, 2)$. The calculation below shows

that the tensor product $V_{(2,1)} \otimes V_{(1,2)}$ contains two copies of $V_{(1,1)}$, while the level 3 fusion product $V_{(2,1)} \otimes_3 V_{(1,2)}$ contains one copy of $V_{(1,1)}$. The information computed by the tensor product and fusion product functions is sufficient to determine the characters of these products, though characters are not implemented in this version of `LieTypes.m2`.

```
i1 : loadPackage("LieTypes");
i2 : sl_3=simpleLieAlgebra("A",2)
o2 = Simple Lie algebra, type A, rank 2
o2 : LieAlgebra
i3 : U=irreducibleLieAlgebraModule({2,1},sl_3);
i4 : V=irreducibleLieAlgebraModule({1,2},sl_3);
i5 : W=irreducibleLieAlgebraModule({1,1},sl_3);
i6 : tensorCoefficient(U,V,W)
o6 = 2
i7 : fusionCoefficient(U,V,W,3)
o7 = 1
```

3. THE CONFORMALBLOCKS.M2 PACKAGE. The `ConformalBlocks.m2` package implements some of Fakhruddin's formulas for conformal block divisors on the moduli space of pointed genus 0 curves $\overline{M}_{0,n}$. Its three main functions compute

(1) the rank of a conformal block bundle,

(2) the intersection number of a conformal block divisor with an F-curve,

(3) the divisor class of the symmetrization of a conformal block divisor.

The version of this package described here uses Macaulay2's combinatorial and linear algebra functions.

Some references for divisors and curves on $\overline{M}_{0,n}$ include [Keel and McKernan 2013; Keel 1992; Arap et al. 2012]. The boundary $\Delta = \partial\overline{M}_{0,n}$ (that is, the locus parametrizing nodal curves) consists of irreducible components Δ_I. These span $\mathrm{Pic}(\overline{M}_{0,n}, \mathbb{Q})$. Moreover, the symmetrizations of the classes Δ_I yield a basis $\{B_2, \ldots, B_{\lfloor n/2 \rfloor}\}$ of $\mathrm{Pic}(\overline{M}_{0,n}, \mathbb{Q})^{S_n}$. The `ConformalBlocks.m2` package implements S_n-symmetric divisors in a new class called `SymmetricDivisorM0nbar`. Divisors may be entered/viewed as linear polynomials in the classes B_i. For instance, the divisor $B_2 + B_3 + 2B_4$ on $\overline{M}_{0,8}$ could be created with the command `symmetricDivisorM0nbar(8,B_2+B_3+2*B_4)`. There are methods, for the `SymmetricDivisorM0nbar` class, for creating and comparing divisors, as well as addition, negation, scalar multiplication, and printing.

We will also be interested in certain combinatorially defined curves in the moduli space called F-curves. These are denoted F_{I_1,I_2,I_3,I_4}, where $I_1 \sqcup I_2 \sqcup I_3 \sqcup I_4$ is a partition of $\{1, \ldots, n\}$ into four nonempty subsets. Averaging such a curve with its S_n translates gives a symmetric curve class; if $\#I_1 = a$, $\#I_2 = b$, $\#I_3 = c$, $\#I_4 = d$, we write $F_{a,b,c,d}$ for this class. The classes $\{F_{j,1,1,n-j-2}\}_{j=1}^{\lfloor n/2 \rfloor - 1}$ form an ordered basis of $H_2(\overline{M}_{0,n}, \mathbb{Q})^{S_n}$.

3.1. ***Ranks of conformal block bundles.*** The function `conformalBlockRank` in `ConformalBlocks.m2` computes ranks of conformal block bundles recursively using propagation and factorization (see [Beauville 1996, Corollary 2.4 and page 84]). We abbreviate $r_{\vec{\lambda}} = \operatorname{rank} \mathbb{V}(\mathfrak{g}, \ell, \vec{\lambda})$ if this will cause no confusion.

In practice, propagation means that if one of the weights is zero, we may drop it. Specifically, let $\vec{\lambda} = (\lambda_1, \ldots, \lambda_n)$, and suppose that $\lambda_n = 0$. Then $\mathbb{V}(\mathfrak{g}, \ell, \vec{\lambda}) = \pi_n^* \mathbb{V}(\mathfrak{g}, \ell, \hat{\lambda})$, where $\hat{\lambda} = (\lambda_1, \ldots, \lambda_{n-1})$ and $\pi_n : \overline{M}_{0,n} \to \overline{M}_{0,n-1}$ is the map forgetting the n-th marked point. In particular, $r_{\vec{\lambda}} = r_{\hat{\lambda}}$.

The factorization rules for conformal block bundles refer to a specific direct sum decomposition of each fiber. We merely state the consequence of factorization for ranks: Let $\vec{\mu} \cup \vec{\nu}$ be a partition of the vector $\vec{\lambda} = (\lambda_1, \ldots, \lambda_n)$ into two vectors, each of length at least 2. Then

$$r_{\vec{\lambda}} = \sum_{\beta \in P_\ell} r_{\vec{\mu} \cup \beta} r_{\vec{\nu} \cup \beta^*}.$$

Here $*$ denotes the involution on the root system given by $-w_0$, where w_0 is the longest word in the Weyl group. Formulas for the action of this involution for the simple Lie algebras are given in [Di Francesco et al. 1997, page 511] and implemented in `LieTypes.m2` with the `starInvolution` function.

To seed the recursion, we must know the ranks of conformal block bundles for n=3. We get these from the fusion coefficients by $\operatorname{rank} \mathbb{V}(\mathfrak{g}, \ell, (\lambda, \mu, \nu)) = N_{\lambda,\mu}^{(\ell)\nu^*}$.

As an example, we compute $\operatorname{rank} \mathbb{V}(\mathfrak{sl}_2, 3, (\omega_1, \ldots, \omega_1))$ on $\overline{M}_{0,8}$:

```
i8 : loadPackage("ConformalBlocks");
i9 : sl_2=simpleLieAlgebra("A",1);
i10 : V=conformalBlockVectorBundle(sl_2,3,{{1},{1},{1},{1},{1},{1},{1},{1}},0);
i11 : conformalBlockRank(V)
o11 = 13
```

3.2. ***Intersection numbers with F-curves.*** Fakhruddin uses factorization to express intersection numbers of $c_1\mathbb{V}(\mathfrak{g}, \ell, \vec{\lambda})$ with an F-curve in terms of degrees of conformal blocks on $\overline{M}_{0,4} \cong \mathbb{P}^1$ and ranks of conformal blocks on $\overline{M}_{0,n'}$ with $n' < n$ [Fakhruddin 2012, Proposition 2.7]. This formula is implemented in the function `FCurveDotConformalBlockDivisor`.

```
i12 : w={{1},{1},{1},{1},{1},{1}};
i13 : V=conformalBlockVectorBundle(sl_2,1,w,0)
o13 = V
o13 : Conformal block vector bundle on M-0-6-bar
i14 : conformalBlockRank(V)
o14 = 1
i15 : FCurveDotConformalBlockDivisor({{1,2,3},{4},{5},{6}},V)
o15 = 1
i16 : FCurveDotConformalBlockDivisor({{1,2},{3,4},{5},{6}},V)
o16 = 0
```

Line o14 tells us that the vector bundle $\mathbb{V}(\mathfrak{sl}_2, 1, (\omega_1, \ldots, \omega_1))$ is a line bundle. The intersection numbers computed in o15 and o16 allow us to give a geometric interpretation of this divisor (see [Alexeev et al. 2014, Theorem 7.2] for details): Let $f : M_{0,6}/S_6 \xrightarrow{\cong} H_2$ be the map which identifies a smooth genus 2 curve with the branch points of its g^1_2. This extends to a map $f : \overline{M}_{0,6}/S_6 \xrightarrow{\cong} \overline{H}_2$ using the theory of admissible covers. By comparing the intersection numbers computed above to those of the pullback $f^*\lambda$ of the λ class on $\overline{M}_2$, we see that $\mathbb{V}(\mathfrak{sl}_2, 1, (\omega_1, \ldots, \omega_1))$ is a multiple of $f^*\lambda$.

3.3. ***Divisor classes of symmetric or symmetrized bundles.*** The S_n-symmetric divisors play an important role in the study of the birational geometry of $\overline{M}_{0,n}$. In addition, they are much easier to study, since $\dim \mathrm{Pic}(\overline{M}_{0,n}, \mathbb{Q}) = 2^{n-1} - \binom{n}{2} - 1$ while $\dim \mathrm{Pic}(\overline{M}_{0,n}, \mathbb{Q})^{S_n} = \lfloor n/2 \rfloor - 1$.

Fakhruddin [2012, Corollary 3.6] gives a formula for computing the symmetrization $\sum_{\sigma \in S_n} c_1\mathbb{V}(\mathfrak{g}, \ell, \sigma\vec{\lambda})$ of a conformal block divisor over its S_n-translates. This is implemented in the function symmetrizedConformalBlockDivisor for an arbitrary n-tuple of weights $\vec{\lambda}$. This function can also be used and is even faster if the set of weights is already S_n-symmetric.

In the example below, we compute $c_1\mathbb{V}(\mathfrak{sl}_6, 1, (\omega_2, \ldots, \omega_2))$ for $n = 6$:

```
i17 : sl_6=simpleLieAlgebra("A",5);
i18 : w2={0,1,0,0,0};
i19 : V=conformalBlockVectorBundle(sl_6,1,apply(6, i -> w2),0);
i20 : D=symmetrizedConformalBlockDivisor(V)
o20 = 288*B  + 864*B
           2        3
o20 : S_6-symmetric divisor on M-0-6-bar
i21 : coefficientList D
o21 = {288, 864}
o21 : List
i22 : coefficientList scale D
o22 = {1, 3}
o22 : List
```

We see that $c_1\mathbb{V}(\mathfrak{sl}_6, 1, (\omega_2, \ldots, \omega_2))$ is a multiple of $B_2 + 3B_3$. The pullback to $\overline{M}_{0,6}$ of the distinguished polarization on the GIT quotient $(\mathbb{P}^1)^6 /\!/ \mathrm{SL}_2$ with the symmetric linearization is also a multiple of $B_2 + 3B_3$; GIT divisors of this form are studied in [Alexeev and Swinarski 2012].

ACKNOWLEDGEMENTS. I am grateful to Valery Alexeev and Angela Gibney, who helped me learn about conformal blocks. I am also grateful to Mike Stillman and Dan Grayson for expert Macaulay2 programming advice, and to Greg Smith and the anonymous referees for several suggestions for improving the packages, their documentation, and this article. I would also like to thank Amelia Taylor and Frank Moore for inviting me to Macaulay2 workshops at Colorado College and

Wake Forest University where this package was completed; these workshops were partially supported by the National Science Foundation under Grant No. 0964128 and the National Security Agency under Grant No. H98230-10-1-0218. This work was also partially supported by the University of Georgia's NSF VIGRE grant DMS-03040000.

SUPPLEMENT. The online supplement contains version 0.5 of LieTypes.m2 and version 2.4 of ConformalBlocks.m2.

REFERENCES.

[Alexeev and Swinarski 2012] V. Alexeev and D. Swinarski, "Nef divisors on $\overline{M}_{0,n}$ from GIT", pp. 1–21 in *Geometry and arithmetic*, edited by C. Faber et al., EMS Ser. Congr. Rep. **8**, Eur. Math. Soc., Zürich, 2012. MR Zbl

[Alexeev et al. 2014] V. Alexeev, A. Gibney, and D. Swinarski, "Higher-level $\mathfrak{sl}_2$ conformal blocks divisors on $\overline{M}_{0,n}$", *Proc. Edinb. Math. Soc.* (2) **57**:1 (2014), 7–30. MR Zbl

[Arap et al. 2012] M. Arap, A. Gibney, J. Stankewicz, and D. Swinarski, "sl_n level 1 conformal blocks divisors on $\overline{M}_{0,n}$", *Int. Math. Res. Not.* **2012**:7 (2012), 1634–1680. MR

[Beauville 1996] A. Beauville, "Conformal blocks, fusion rules and the Verlinde formula", pp. 75–96 in *Proceedings of the Hirzebruch 65 Conference on Algebraic Geometry* (Ramat Gan, 1993), edited by M. Teicher, Israel Math. Conf. Proc. **9**, Bar-Ilan Univ., Ramat Gan, 1996. MR Zbl

[Di Francesco et al. 1997] P. Di Francesco, P. Mathieu, and D. Sénéchal, *Conformal field theory*, Springer, 1997. MR Zbl

[Fakhruddin 2012] N. Fakhruddin, "Chern classes of conformal blocks", pp. 145–176 in *Compact moduli spaces and vector bundles*, edited by V. Alexeev et al., Contemp. Math. **564**, Amer. Math. Soc., Providence, RI, 2012. MR Zbl

[Keel 1992] S. Keel, "Intersection theory of moduli space of stable n-pointed curves of genus zero", *Trans. Amer. Math. Soc.* **330**:2 (1992), 545–574. MR Zbl

[Keel and McKernan 2013] S. Keel and J. McKernan, "Contractible extremal rays on $\overline{M}_{0,n}$", pp. 115–130 in *Handbook of moduli: Vol. II*, edited by G. Farkas and I. Morrison, Adv. Lect. Math. (ALM) **25**, International Press, Somerville, MA, 2013. MR Zbl

[Ueno 2008] K. Ueno, *Conformal field theory with gauge symmetry*, Fields Institute Monographs **24**, American Mathematical Society, Providence, RI, 2008. MR Zbl

RECEIVED: 16 Feb 2014 REVISED: 23 Jun 2018 ACCEPTED: 2 Aug 2018

DAVID SWINARSKI:

dswinarski@fordham.edu

Department of Mathematics, Fordham University, New York, United States

JSAG **8** (2018), 87–94
dx.doi.org/10.2140/jsag.2018.8.87

The Journal of Software for
Algebra and Geometry

Divisor Package for Macaulay2

KARL SCHWEDE AND ZHAONING YANG

ABSTRACT: This note describes a Macaulay2 package for handling divisors. Group operations for divisors are included. There are methods for converting divisors to reflexive or invertible sheaves. Additionally, there are methods for checking whether divisors are Cartier, $\mathbb{Q}$-Cartier, simple normal crossings, or generate base point free linear systems, or satisfy numerous other conditions.

1. INTRODUCTION. Divisors are fundamental objects of study within algebraic geometry and commutative algebra. In the `Divisor.m2` package for [Macaulay2], we provide a wrapper object for studying Weil and Cartier divisors. We include tools for studying divisors on both affine and projective varieties.

In this package, divisors are stored (roughly) as formal linear combinations of height-1 prime ideals, with coefficients from $\mathbb{Z}$, $\mathbb{Q}$, or $\mathbb{R}$. We include group and scaling operations for divisors, as well as various methods for constructing modules $\mathcal{O}_X(D)$ from divisors D (and vice versa). We also include code for determining whether divisors are linearly or $\mathbb{Q}$-linearly equivalent, and for checking whether divisors are Cartier or $\mathbb{Q}$-Cartier (or finding the non-Cartier locus). Finally, we also include a number of functions for handling reflexive modules, ideals and their powers.

We realize there is a `Divisor` class defined in a tutorial in the Macaulay2 help system. In that implementation, divisors are given as a pair of ideals — an ideal corresponding to the positive part and an ideal corresponding to the negative part. Our approach offers the advantage that it is easier for the user to see the structure of the divisor. Additionally, certain operations are much faster in our approach.

We warn the user that when a divisor is created, Gröbner bases are constructed for each prime ideal defining a component of the divisor. Hence, the construction phase may be slower than other potential implementations (and in fact slower than our initial implementation). However, we feel that this choice offers advantages of

Schwede was supported in part by the NSF FRG Grant DMS #1265261/1501115, NSF CAREER Grant DMS #1252860/1501102, NSF Grant #1801849 and a Sloan Fellowship. Yang was supported in part by the NSF CAREER Grant DMS #1252860/1501102.
MSC2010: primary 14C20; secondary 13B22.
Keywords: divisors, reflexive modules, Macaulay2.
Divisor.m2 version 0.3

execution speed for several functions as well as substantial improvements in code readability.

Within the package, it is tacitly assumed that the ambient ring on which we are working is normal. This includes the projective case, so care should be taken to make sure the graded ring you are working on satisfies Serre's second condition, see for example [Hartshorne 1977, Theorem 8.22A] or [Bruns and Herzog 1993, Proposition 2.2.21]. While one can talk about subvarieties of codimension 1 on more general schemes, the correspondence between divisors and reflexive sheaves is much more complicated, so we restrict ourselves to the normal case. For an introduction to the theory of rank-1-reflexive sheaves on "nice" schemes, see [Hartshorne 1994; 2007]; and for a more basic introduction see, for instance, [Hartshorne 1977, Chapter II, Sections 5–7].

This paper is structured as follows. We first give a brief introduction to the construction, conversion, and group operation functions in Section 2. We then discuss the methods for converting divisors D to modules $\mathcal{O}_X(D)$ and converting modules back to divisors in Section 3. Section 4 describes how to determine if divisors satisfy various properties (for instance `isCartier` or `isSNC`). We conclude with a section on future plans.

2. CONSTRUCTION, CONVERSION AND GROUP OPERATIONS FOR DIVISORS. This package includes a number of ways to construct a divisor (an object of class `WeilDivisor`), illustrated here.

```
i1 : needsPackage "Divisor";
i2 : R = QQ[x,y,u,v]/ideal(x*y-u*v);
i3 : D = divisor({2, 3}, {ideal(x,u), ideal(x, v)})
o3 = 3*Div(x, v) + 2*Div(x, u)
o3 : WeilDivisor on R
i4 : E = divisor(x)
o4 = Div(u, x) + Div(v, x)
o4 : WeilDivisor on R
i5 : F = divisor( (ideal(x,u))^2*(ideal(x,v))^3 )
o5 = 3*Div(v, x) + 2*Div(u, x)
o5 : WeilDivisor on R
```

The output is a formal sum of height-1 prime ideals. The first method requires a list of integers and a list of prime ideals. The third construction method finds a divisor defined by the given ideal in codimension 1.

We have different classes for $\mathbb{Q}$-divisors and $\mathbb{R}$-divisors (`QWeilDivisor` and `RWeilDivisor` respectively); these are constructed via the `divisor` function with the `CoeffType =>` option set or by multiplying a `WeilDivisor` by a rational or real number. See the documentation.

All types of divisors are ancestors of the `HashTable` class. Internally, they are hash tables where each key is a list of Gröbner basis generators for a prime height-1

ideal and each associated value is a list, the first entry of which is the coefficient of the prime divisor and the second entry is the prime ideal used to display the divisor (it tries to match how the user entered it for ease of reading). Besides the keys corresponding prime divisors, there is a key that specifies the ambient ring and another key that points to a `CacheTable`.

One can convert one type of divisor to another more general class, either by multiplication by appropriate coefficients or by calling appropriate functions.

```
i2 : R = QQ[x,y,u,v]/ideal(x*y-u*v);
i3 : D = divisor({1, -3}, {ideal(x,u), ideal(y,u)});
o3 : WeilDivisor on R
i4 : 1/1*D
o4 = -3*Div(y, u) + Div(x, u)
o4 : QWeilDivisor on R
i5 : toQWeilDivisor(D)
o5 = Div(x, u) + -3*Div(y, u)
o5 : QWeilDivisor on R
```

One can convert $\mathbb{Q}$ or $\mathbb{R}$-divisors back to Weil divisors as follows.

```
i3 : D = divisor( {2/3, -1/2}, {ideal(x,u), ideal(y, v)}, CoeffType=>QQ)
o3 = 2/3*Div(x, u) + -1/2*Div(y, v) of R
o3 : QDiv
i4 : isWDiv(D)
o4 = false
i5 : isWDiv(6*D)
o5 = true
i6 : toWDiv(6*D)
o6 = 4*Div(x, u) + -3*Div(y, v) of R
o6 : WDiv
```

See the documentation for more examples. Alternatively, the functions `ceiling` and `floor` will convert any $\mathbb{Q}$ or $\mathbb{R}$-divisor to a Weil divisor by taking the ceiling or floor of the coefficients, respectively. More generally, one can call the method `applyToCoefficients` to apply any function to the coefficients of a divisor (since divisors are a type of `HashTable`, this is just done via the `applyValues` function).

Divisors form an abelian group and one can add `WeilDivisor/QWeilDivisor/RWeilDivisor` to each other to obtain new divisors. Likewise one can scale by integers, rational numbers or real numbers.

```
i3 : D = divisor({1, -2}, {ideal(x,u), ideal(x, v)}); E = divisor(u);
o3 : WeilDivisor on R
o4 : WeilDivisor on R
i5 : 3*D+E
o5 = 4*Div(x, u) + -6*Div(x, v) + Div(u, y)
o5 : WeilDivisor on R
i6 : D - (1/2)*E
o6 = -2*Div(x, v) + 1/2*Div(x, u) + -1/2*Div(u, y)
o6 : QWeilDivisor on R
```

Since divisors are implemented as subclasses of hash tables, these operations are easily executed internally via the `merge` and `applyValues` commands.

3. MODULES, IDEALS, DIVISORS AND APPLICATIONS. It is well known that divisors are so useful because of their connections with invertible and reflexive sheaves. This package includes many functions for conversion between these types of objects. For instance, we have the following:

```
i2 : R = QQ[x,y,z]/ideal(x*y-z^2); needsPackage "Divisor";
i3 : D = divisor(ideal(x, z));
o3 : WeilDivisor on R
i4 : OO(D)
o4 = image {-1} | x z |
           {-1} | z y |
o4 : R-module, submodule of R
i5 : divisor(o4)
o5 = -Div(z, x)
o5 : WeilDivisor on R
i6 : divisor(o4, IsGraded=>true)
o6 = Div(z, x)
o6 : WeilDivisor on R
```

The function `OO` produces a module M so that $\widetilde{M} \cong \mathcal{O}_X(D)$ (and the gradings of M are set appropriately). The function `divisor(M)` only produces a divisor E such that $\mathcal{O}_X(E)$ is isomorphic to $\widetilde{M}$. In particular, `divisor(OO(`D`))` will only produce a divisor linearly equivalent to D.

We use a straightforward strategy to compute `OO(D)`. If `D` $= \sum_{i=1}^{m} a_i P_i$ where the a_i are integers and the P_i are primes, then we can compute $\bigotimes P_i^{-a_i}$ (keeping in mind negative exponents mean applying $\mathrm{Hom}_R(-, R)$) and compute the reflexification (see the method `reflexify`). We do several things to make this computation faster. Firstly, we break up the divisor into the positive and negative parts, and handle them separately (applying the `reflexify` method as little as possible). Then, instead of computing $P_i^{|a_i|}$, which can have many generators, we form an ideal generated by the generators of P_i raised to the $|a_i|$-th powers. Since this agrees with $P_i^{|a_i|}$ in codimension 1, it will give the correct answer up to reflexification. We have noticed substantial speed improvements using this technique.

The function `divisor(Module)` works as follows. First, it embeds the module as an ideal $I \subseteq R$ via the function `embedAsIdeal`. After we have an ideal I, we call `divisor(I)`. This finds a divisor D such that $\mathcal{O}_X(D)$ is isomorphic to the given ideal I (in a nongraded sense). The function `divisor(Ideal)` does this by looking at the minimal height-1 primes Q_i of the ideal I and finding the maximum power n_i such that $I \subseteq Q_i^{(n_i)}$ (the symbolic power). Note that because Q_i has height 1, we know $Q_i^{(n_i)} = (Q_i^{n_i})^{**}$ where $-^{**}$ denotes reflexification/S2-ification of the ideal. Finding this maximal power is done by a binary search. Again, for speed, we compute $(Q_i^{n_i})^{**}$ as $(Q_i^{\lfloor n_i \rfloor})^{**}$. If the `IsGraded` flag is set to `true`,

`divisor(Module)` corrects the degree of the divisor by adding or subtracting the divisor of an element of appropriate degree (you can see this being done in the example above). Finding the element of appropriate degree is accomplished via the function `findElementOfDegree`, which uses Smith normal form in the multidegree setting to solve the system of linear diophantine equations and find a monomial of the given multidegree.

Remark 3.1. *A variant of the function* `embedAsIdeal` *appeared in the Macaulay*2 *documentation in the* `Divisor` *tutorial*; *it also appeared in the work of Moty Katzman. Our version is slightly more robust than those as it tries to embed the module into the ring in several ways*, *including some random attempts* (*see the documentation for how to control the number of random attempts*).

Instead of calling `divisor(Module)`, one can call `divisor(Module, Section => f)`. This function finds the unique effective divisor D corresponding to a global section $f \in M$ of our module. The function `divisor(Ideal, Section => f)` behaves similarly. The strategy is the same as above, and additionally one tracks the section and adds a divisor corresponding to the section at the end.

It is worth mentioning that the function `canonicalDivisor` simply computes the canonical module via an appropriate Ext and then calls `divisor(Module)`. If you wish to construct a canonical divisor on a projective variety, make sure to set the `IsGraded` option to `true`.

Pulling back divisors. Utilizing the module and divisor correspondence `pullBack` pulls back a divisor along a map Spec $S \to$ Spec R induced by a ring map $R \to S$. The user has a choice of two algorithms built into this function. The first works for nearly any map, provided that the divisor is Cartier, and it also works for arbitrary divisors in the flat or finite case. The second, which is the default strategy, only gives accurate answers if the map is flat, or if the map is finite (or if the prime components of the divisor are Cartier). It can be faster than the first algorithm, especially for divisors with large coefficients. To use the first algorithm, use `Strategy => Sheaves`, to use the second, use `Strategy => Primes`.

Let us briefly describe these two strategies. The first algorithm pulls back the sheaf $\mathcal{O}(D)$, keeping track of a section appropriately. The second algorithm extends each prime ideal defining a prime divisor of D to an ideal of S, then it calls `divisor(Ideal)` on each such ideal and sums them keeping track of coefficients appropriately.

Consider the following example where we look at pulling back a divisor after blowing up the origin (we only consider one chart of the blowup).

```
i2 : R = QQ[x,y];
i3 : S = QQ[a,b];
i4 : f = map(S, R, {a*b, b});
o4 : RingMap S <--- R
```

```
i5 : D = divisor(x*y*(x+y)*(x-y))
o5 = Div(x+y) + Div(-x+y) + Div(x) + Div(y)
o5 : WeilDivisor on R
i6 :  pullback(f, D)
o6 = Div(a+1) + Div(a-1) + 4*Div(b) + Div(a)
o6 : WeilDivisor on S
```

Note one of the components was lost in this pull-back, as it should have been. The coefficient of the exceptional divisor is also 4, as it should be.

Global sections. There are only a few built-in functions for dealing with global sections of modules corresponding to divisors in the current version (in the future we hope to add more tools to do this). Of course, the user may call `basis(0, OO(D))` to get the global sections of a module corresponding to a divisor. In this section, we describe briefly two functions for handling global properties of divisors.

The function `mapToProjectiveSpace` gets the global sections of $\mathcal{O}(D)$ and then computes the corresponding map to projective space. This of course assumes the divisor is graded. In the example below we project $\mathbb{P}^1 \times \mathbb{P}^1$ to one of its terms by calling `mapToProjectiveSpace` along a divisor of one of the rulings.

```
i2 : R = QQ[x,y,u,v]/ideal(x*y-u*v);
i3 : D = divisor(ideal(x,u));
o3 : WeilDivisor on R
i4 : mapToProjectiveSpace(D)
o4 = map(R,QQ[YY , YY ],{v, x})
                 1    2
o4 : RingMap R <--- QQ[YY , YY ]
                           1    2
```

Still assuming the divisor is graded, the function `baseLocus` finds a defining ideal for the locus where $\mathcal{O}(D)$ is *not* generated by global sections. This is done by computing the cokernel of $\mathcal{O}^{\oplus n} \to \mathcal{O}(D)$ where $H^0(X, \mathcal{O}(D))$ has a basis of n distinct global sections and the map is the obvious one. In the following example, we compute the base locus of a point on an elliptic curve, and also two times a point on an elliptic curve (which is degree 2 and hence base point free).

```
i2 : R = QQ[x,y,z]/ideal(y^2*z-x*(x+z)*(x-z));
i3 : D = divisor( ideal(x,y) );
o3 : WeilDivisor on R
i4 : baseLocus(D)
o4 = ideal (y, x)
o4 : Ideal of R
i5 : baseLocus(2*D)
o5 = ideal 1
o5 : Ideal of R
```

4. CHECKING PROPERTIES OF DIVISORS. The package `Divisor.m2` can check divisors for several properties. First, we describe the method `isCartier`.

```
i2 : R = QQ[x,y,z]/ideal(x^2-y*z);
i3 : D = divisor(ideal(x,y));
i4 : isCartier(D)
o4 = false
i5 : nonCartierLocus(D)
o5 = ideal (z, y, x)
o5 : Ideal of R
i6 : isCartier(2*D)
o6 = true
i7 : isCartier(D, IsGraded => true)
o7 = true
```

The algorithm behind this function is as follows. We compute $\mathcal{O}_X(-D)\cdot\mathcal{O}_X(D)$ and check if it is equal to $\mathcal{O}_X$. In general, $\mathcal{O}_X(-D)\cdot\mathcal{O}_X(D)$ always defines an ideal defining the non-Cartier locus of D, hence the command `nonCartierLocus`. If the option `IsGraded => true`, then the relevant functions saturate the ideals with respect to the irrelevant ideal.

We also briefly describe the method `isQCartier`.

```
i8 : isQCartier(5, D)
o8 = 2
```

This checks whether any multiples $n \cdot D$ of a Weil divisor or $\mathbb{Q}$-divisor D are Cartier for any integer n less than or equal to the first argument (in this case $n \leq 5$). It may actually search a little higher than the first argument in the $\mathbb{Q}$-Cartier case due to rounding issues. If it finds that nD is Cartier, it returns the integer n. If it doesn't find any Cartier divisors, it returns `0`.

Some other useful functions include `isPrincipal` and `isLinearEquivalent`. Checking whether a divisor is principal just comes down to checking whether $\mathcal{O}_X(D)$ is a free module and checking whether $D \sim E$ just boils down to checking whether $D - E$ is principal. In the graded case, we can do this via Macaulay2 using the `prune` and `isFreeModule` commands. Unfortunately, we do not know an algorithm for deciding if a nongraded module is free (although we still try to prune the module and more). Therefore `isPrincipal` and `isLinearEquivalent` can give a false negative for non-graded divisors (the function warns you when this is the case). In the same way, the option `IsGraded` can be applied within `isLinearEquivalent`, which checks that $\mathcal{O}_X(D - E)$ is principal of degree zero.

We can also check whether a divisor D has simple normal crossings by calling `isSNC`. This first checks that the ambient space of D is regular, then it checks that each prime divisor of D defines a regular scheme, and finally it checks that every intersection of prime divisors of D also defines a regular scheme of the appropriate dimension.

5. FUTURE PLANS. There are a number of ways that this package should be expanded. One of the most important things to be done is to further develop the

global methods related to divisors. We have recently added the ability to check whether a divisor is very ample via the `isVeryAmple` function, which uses the `RationalMaps` package. However, there is much more to be done. Some basic intersection theory between divisors and smooth curves would be natural to include.

While the latest version of the package stores the outputs of some functions in the cache, this can still be improved. For example, there are likely ways to take advantage of knowing that a given divisor is Cartier or $\mathbb{Q}$-Cartier.

ACKNOWLEDGEMENTS. We thank Tommaso de Fernex, David Eisenbud, Daniel Grayson, Anurag Singh, Greg Smith, Mike Stillman, and the referees for useful conversations and comments on the development of this package. We also thank the referee for numerous useful comments on this paper.

SUPPLEMENT. The online supplement contains version 0.3 of `Divisor.m2`.

REFERENCES.

[Bruns and Herzog 1993] W. Bruns and J. Herzog, *Cohen–Macaulay rings*, Cambridge Studies in Advanced Mathematics **39**, Cambridge University Press, 1993. MR

[Hartshorne 1977] R. Hartshorne, *Algebraic geometry*, Graduate Texts in Mathematics **52**, Springer, 1977. MR Zbl

[Hartshorne 1994] R. Hartshorne, "Generalized divisors on Gorenstein schemes", *K-Theory* **8**:3 (1994), 287–339. MR Zbl

[Hartshorne 2007] R. Hartshorne, "Generalized divisors and biliaison", *Illinois J. Math.* **51**:1 (2007), 83–98. MR Zbl

[Macaulay2] D. R. Grayson and M. E. Stillman, "Macaulay2: a software system for research in algebraic geometry", Available at http://www.math.uiuc.edu/Macaulay2.

RECEIVED: 31 Oct 2014 REVISED: 15 Jul 2018 ACCEPTED: 31 Aug 2018

KARL SCHWEDE:

schwede@math.utah.edu

Department of Mathematics, University of Utah, Salt Lake City, UT, United States

ZHAONING YANG:

zyy5054@gmail.com

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

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