J. Symbolic Computation (1998) **26**, 409–431
Article No. sy980221

Strategies for Computing Minimal Free Resolutions[†]

ROBERTO LA SCALA[‡¶] AND MICHAEL STILLMAN[§‖]

[‡] *Dipartimento di Matematica, Università di Bari, Italy*
[§] *Department of Mathematics, Cornell University, U.S.A.*

In the present paper we study algorithms based on the theory of Gröbner bases for computing free resolutions of modules over polynomial rings. We propose a technique which consists in the application of special selection strategies to the Schreyer algorithm. The resulting algorithm is efficient and, in the graded case, allows a straightforward minimalization algorithm. These techniques generalize to factor rings, skew commutative rings, and some non-commutative rings. Finally, the proposed approach is compared with other algorithms by means of an implementation developed in the new system Macaulay2.

1. Introduction

One of the most important computations in algebraic geometry or commutative algebra that a computer algebra system should provide is the computation of finite free resolutions of ideals and modules. Resolutions are used as an aid to understand the subtle nature of modules and are also a basis of further computations, such as computing sheaf cohomology, local cohomology, Ext, Tor, etc. Modern methods for calculating free resolutions derive from the theory of Gröbner bases. These methods were introduced at the end of the 1970s by Richman (1974); Spear (1977); Schreyer (1980) and have survived in computer algebra systems up to now. However, the problem with these algorithms is that many computations of interest for researchers were out of range. This is giving impulse to authors such as Capani *et al.* (1997), Siebert (1996) and ourselves to develop decisive improvements of the resolution techniques.

Resolution algorithms based on Gröbner bases can be divided essentially into two types. The first type is based on computing the syzygy module on a minimal set of generators. The second type, initially used by Frank Schreyer, is based on computing the syzygy module on a Gröbner basis. In both cases, using induced term orderings leads to a large improvement in the sizes of the Gröbner bases involved. Which of these two methods is best depends in part on the specific input ideal or module. However, we have found that for problems of interest, the Schreyer technique, together with the improvements that we suggest, on the average outperforms the other methods.

[†]This research was performed with the contribution of MURST, and M. Stillman would like to thank the NSF for partial support during the preparation of this manuscript.
[¶]E-mail: lascala@dm.uniba.it
[‖]E-mail: mike@math.cornell.edu

0747–7171/98/100409 + 23 30.00/0

The syzygies of the Schreyer resolution are usually computed level by level. (i.e., first syzygies first, then second syzygies, etc.) In this way just one syzygy is obtained by any S-polynomial reduction. Our improvement of the Schreyer algorithm is based essentially on the remark that we can use global strategies instead for the selection of the S-polynomials which allow the computation of couples of syzygies from single reductions. This results not only in an optimization of the calculation of the Schreyer resolution but also of its minimalization. It can be proved in fact that with respect to suitable strategies a minimal resolution corresponds exactly to the syzygies traced by S-polynomial reductions to zero. It follows that using our technique the Betti numbers are derived by the Schreyer resolution with no additional computations.

An important feature of the proposed algorithms is that they can be easily extended. In the present paper we explain how to generalize them to factor rings and show that the resolution procedure can be applied to non- graded ideals and modules. We describe an implementation, and some of the choices one has when computing a resolution. We suggest some optimizations that we have found to be useful. Finally, we compare these algorithms with other algorithms, using a suite of examples. These tests are performed using an installation developed in the new system Macaulay2 (Grayson and Stillman, 1993–1998).

2. Preliminaries

In this section, we define our notation regarding Gröbner bases. Since the algorithms we describe also work over a factor ring R of a polynomial ring S, we also describe our notation regarding Gröbner bases in this setting. Without too much more difficulty, one could extend these definitions to more general situations, e.g. (factor rings of) skew-commutative polynomial rings, Weyl algebras, and non- commutative polynomial rings. The algorithms given in this paper would work, with small modifications, in these more general settings.

Let $S = K[x_1, \ldots, x_n]$ be a polynomial ring over the field K, endowed with a term order $>$ that we fix once and for all.

A *power product* (or monomial of S) is an element

$$t = x_1^{\alpha_1} \cdots x_n^{\alpha_n} \in S,$$

where $\alpha_1, \ldots, \alpha_n$ are non-negative integers.

Any non-zero element $f \in S$ may be written uniquely as the sum

$$f = c_1 \cdot m_1 + \cdots + c_k \cdot m_k,$$

where $0 \neq c_i \in K$, m_i monomials, and $m_1 > m_2 > \cdots > m_k$. We define:

$\mathrm{lc}(f) = c_1$, the *leading coefficient*,
$\mathrm{lm}(f) = m_1$, the *leading monomial*.

For any set $G \subset S$, we let $\mathrm{in}(G)$ denote the K-vector space spanned by the monomials $\{\mathrm{lm}(f) : f \in G\}$. If $J \subset S$ is an ideal, then $\mathrm{in}(J) \subset S$ is the monomial ideal generated by the lead terms of J.

Let $R = S/J$ be a factor ring of S. We may extend the above notions to this situation. Let $N \subset S$ be the K-vector space spanned by the set of standard monomials of R, that

is, the set of monomials of S not in $\text{in}(J)$. Any element $f \in R$ may be uniquely written as the image of an element $g \in N$, and we set $\text{lc}(f) = \text{lc}(g)$ and $\text{lm}(f) = \text{lm}(g) \in N$.

For any set $G \subset R$, let $\text{in}(G)$ denote the K-vector space generated by the lead monomials $\{\text{lm}(f) : f \in G\}$. Thus, $N = \text{in}(R)$, and $\text{in}(S) = N \oplus \text{in}(J)$, as K-vector spaces.

Let F be a free module over R, and let $\hat{F}$ be the free S-module with the same rank as F and corresponding basis. A *monomial* of F is by definition any element

$$m = t \cdot e,$$

where $t \in N$ is a standard monomial and e is any element of the canonical basis of $\hat{F}$.

A *term order* on F is a total order on the monomials of F such that:

(i) if $m < n$, then $t \cdot m < t \cdot n$;
(ii) if $s < t$, then $s \cdot e < t \cdot e$

for all m, n monomials of F, s, t power products in S, and e any basis element of F.

Fix a term order on F. Then, any element $f \in F$ may be written uniquely (as the image of an element) in the form:

$$f = c_1 \cdot m_1 + \cdots + c_k \cdot m_k,$$

where $0 \neq c_i \in K$, m_i monomials, and $m_1 > m_2 > \ldots > m_k$. If $m_1 = t \cdot e$, for the module element f we define:

$\text{lc}(f) = c_1$, the *leading coefficient*,
$\text{lm}(f) = m_1$, the *leading monomial*,
$\text{lpp}(f) = t$, the *leading power product*.

As usual, for any $G \subset F$, we denote by $\text{in}(G) \subset \hat{F}$ the K-vector space generated by $\{\text{lm}(f) : f \in G\}$.

We say that $\{g_1, \ldots, g_s\} \subset I \subset F$ is a *Gröbner basis* of the R-module I, if $\{\text{lm}(g_1), \ldots, \text{lm}(g_s)\}$ generates $\text{in}(I)$, that is, every monomial in $\text{in}(I)$ is divisible by some $\text{lm}(g_i)$. The Gröbner basis is called *auto-reduced* if every lead monomial $\text{lm}(g_i)$ divides no monomial occuring in any g_j, other than itself. The Gröbner basis is called *irredundant* if $\text{in}(I)$ is minimally generated by $\{\text{lm}(g_1), \ldots, \text{lm}(g_s)\}$, which in turn means that this set generates $\text{in}(I)$, and no lead term $\text{lm}(g_i)$ divides any $\text{lm}(g_j)$, for $j \neq i$.

3. The Schreyer Resolution and Its Frame

Throughout this section, $R = S/J$ is a factor ring of the polynomial ring S, and $M = F_0/I$ is an R-module, where F_0 is a free module. If both M and R are graded, then all of our sequences of modules, and resolutions will be graded as well.

Consider a sequence of R-homomorphisms:

$$\Phi : \cdots \longrightarrow F_l \xrightarrow{\varphi_l} F_{l-1} \xrightarrow{\varphi_{l-1}} \cdots \xrightarrow{\varphi_2} F_1 \xrightarrow{\varphi_1} F_0$$

where each F_i is a free R-module with a given (canonical) basis $\mathcal{E}_i$. Let $\mathcal{C}_i = \varphi_i(\mathcal{E}_i)$ be the image of the given basis, and define the level of an element f in $\mathcal{C}_i$ to be $\text{lev}(f) = i$. Note that Φ is not necessarily a complex.

Definition 3.1. Let $\tau = \{\tau_i\}$ be a sequence of term orderings τ_i on the F_i. We call τ a *term ordering* on Φ if it satisfies the following compatibility relationship:

$$s \cdot e_1 < t \cdot e_2 \text{ whenever } s \cdot \operatorname{lm} \varphi_i(e_1) < t \cdot \operatorname{lm} \varphi_i(e_2),$$

where e_1 and e_2 are elements of $\mathcal{E}_i$.

DEFINITION 3.2. Given a Φ as above, and a term ordering on Φ, define the *initial terms* of Φ, $\operatorname{in}(\Phi)$, to be the sequence of (graded) R-homomorphisms:

$$\Xi = \operatorname{in}(\Phi) : \cdots \longrightarrow F_l \xrightarrow{\xi_l} F_{l-1} \xrightarrow{\xi_{l-1}} \cdots \xrightarrow{\xi_2} F_1 \xrightarrow{\xi_1} F_0$$

where $\xi_i(e) = \operatorname{lm} \varphi_i(e)$, for all e in $\mathcal{E}_i$. That is, $\operatorname{in}(\Phi)$ consists of the leading monomials of the columns of (each matrix of) Φ.

Notice that a term ordering on Φ is also a term ordering for Ξ. The converse holds, if for this order, $\Xi = \operatorname{in}(\Phi)$.

DEFINITION 3.3. A *Schreyer resolution* of an R-module $M = F_0/I$ is an exact sequence:

$$\Phi : \cdots \to F_l \xrightarrow{\varphi_l} F_{l-1} \xrightarrow{\varphi_{l-1}} \cdots \xrightarrow{\varphi_2} F_1 \xrightarrow{\varphi_1} F_0$$

together with a term ordering on Φ, such that:

(i) $\operatorname{coker}(\varphi_1) = M$;
(ii) $\varphi_i(\mathcal{E}_i)$ forms an irredundant Gröbner basis of $\operatorname{image}(\varphi_i)$ (for all i where $F_i \neq 0$);

Our plan in the next section is to give an algorithm that computes a Schreyer resolution. A useful way of picturing this resolution is by its "frame".

DEFINITION 3.4. A *Schreyer frame* of $M = F_0/I$ is a sequence of (graded) R-homomorphisms:

$$\Xi : \cdots \to F_l \xrightarrow{\xi_l} F_{l-1} \xrightarrow{\xi_{l-1}} \cdots \xrightarrow{\xi_2} F_1 \xrightarrow{\xi_1} F_0$$

where each column is a monomial, and a term ordering on Ξ such that:

(i) $\xi_1(\mathcal{E}_1)$ is a minimal set of generators for $\operatorname{in}(I)$;
(ii) $\xi_i(\mathcal{E}_i)$ is a minimal set of generators for $\operatorname{in}(\ker \xi_{i-1})$ $(i \geq 2)$.

If Φ is a Schreyer resolution, then $\operatorname{in}(\Phi)$ is a Schreyer frame. The reason we introduce this concept is because one can compute a Schreyer frame first, and then "fill it in" to form a Schreyer resolution.

Let Ξ be a Schreyer frame. It is useful to introduce the following notation. Given i, and a basis element $e \in \mathcal{E}_{i-1}$, we put:

$$\mathcal{B}_i = \xi_i(\mathcal{E}_i);$$
$$\mathcal{E}_i(e) = \{\epsilon \in \mathcal{E}_i \ : \ \xi_i(\epsilon) = s \cdot e, \text{ for some power product } s\}.$$

Thus $\mathcal{E}_i(e)$ consists of those basis elements that map to multiples of e. Note that this set can be empty.

The following lemma is the basis of our algorithm to compute a Schreyer frame. Let $\xi_i : F_i \to F_{i-1}$ be a map of free R-modules, such that each element $\xi_i(\epsilon)$ is a monomial, for all $\epsilon \in \xi_i$. Suppose that ξ_i is endowed with a term ordering.

LEMMA 3.5. *Given the above setup, we have that* $\operatorname{in}(\ker \xi_i)$ *is minimally generated by*

$$\bigcup_{e \in \mathcal{E}_{i-1}} \bigcup_{j=2}^{r} \operatorname{mingens}\left((\operatorname{in}(J), t_1, \ldots, t_{j-1}) : t_j\right) \cdot \epsilon_j$$

where for each e *in the outer union, if* $\mathcal{E}_i(e) = \{\epsilon_1, \ldots, \epsilon_r\}$ *then* $\xi_i(\epsilon_j) = t_j \cdot e$, *and* mingens *defines the subset of the minimal generators of the considered monomial ideal which do not lie in* $\operatorname{in}(J)$.

PROOF. Let $\sum_j g_j \cdot m_j = 0$ where m_j are minimal generators of $\operatorname{in}(\operatorname{image} \xi_i)$ and g_j are standard polynomials of S. Let $s_k \cdot \epsilon_k$ be the leading monomial of the corresponding syzygy with $s_k = \operatorname{lm}(g_k)$. For cancelling $s_k \cdot m_k$ in the sum there are exactly two possibilities:

$s_k \cdot m_k = s_h \cdot m_h$ for some h,
$s_k \cdot t_k$ belongs to $\operatorname{in}(J)$,

where $m_k = t_k \cdot e$.

This lemma immediately translates into an algorithm to compute a Schreyer frame (or all possible frames) given $\operatorname{in}(J)$ and $\operatorname{in}(I)$.

PROPOSITION 3.6. *If* Ξ *is a Schreyer frame for* M, *then there exists a Schreyer resolution* Φ *such that* $\Xi = \operatorname{in}(\Phi)$.

PROOF. With respect to the term ordering assigned on the free module F_0, any irredundant Gröbner basis $\mathcal{C}_1$ of the submodule $I \subset F_0$ satisfies $\operatorname{in}(\mathcal{C}_1) = \mathcal{B}_1$. Note now that for any monomial $m \in \mathcal{B}_2$, there exists an element g in $\mathcal{C}_1 = \varphi_1(\mathcal{E}_1)$ such that:

$$m = t \cdot \epsilon,$$

where t is a standard power product, $\epsilon \in \mathcal{E}_1$ and $\varphi_1(\epsilon) = g$. By definition, the set $\mathcal{C}_2 = \varphi_2(\mathcal{E}_2)$ is formed by the syzygies traced by the reductions to zero of the elements $t \cdot g$ as m varies in $\mathcal{B}_2$. Since the term ordering of F_1 satisfies the condition of Definition 3.1, we have:

$$\operatorname{in}(\ker \xi_1) = \operatorname{in}(\ker \varphi_1).$$

It follows that $\mathcal{C}_2$ is an irredundant Gröbner basis of $\ker(\varphi_1)$ and $\operatorname{in}(\mathcal{C}_2) = \mathcal{B}_2$. Iterating for all the levels i, we get a Schreyer resolution Φ such that $\Xi = \operatorname{in}(\Phi)$.

Note that this is essentially Schreyer's original proof that the syzygy module is constructed from a Gröbner basis computation. The resolution Φ produced in the above proposition is not quite unique, given the Schreyer frame Ξ. One could make it unique by requiring that each $\varphi_i(\mathcal{E}_i)$ forms an auto-reduced Gröbner basis.

PROPOSITION 3.7. *If* Φ *is a complex of free* S*-modules, with a term ordering, such that* $\operatorname{in}(\Phi)$ *is a Schreyer frame of* M, *then* Φ *is a free resolution of* M.

PROOF. Clear.

The following proposition explains how a term ordering can be assigned on a frame.

PROPOSITION 3.8. *Let* $\Xi : \cdots \longrightarrow F_l \xrightarrow{\xi_l} F_{l-1} \xrightarrow{\xi_{l-1}} \cdots \xrightarrow{\xi_2} F_1 \xrightarrow{\xi_1} F_0$ *be a sequence of homomorphisms where each column is a monomial. To give a term ordering on* Ξ *is equivalent to give a term ordering on* F_0 *and place total orders on the sets* $\mathcal{E}_i(e) \neq \emptyset$*, for every* $i \geq 1$ *and* $e \in \mathcal{E}_{i-1}$.

PROOF. By induction on the level i, the term ordering on the free module F_i can be defined as follows. For any s, t power products, $\epsilon_1, \epsilon_2 \in \mathcal{E}_i$, $m = \xi_i(\epsilon_1), n = \xi_i(\epsilon_2)$, we put:

$$\begin{array}{l} s \cdot \epsilon_1 < t \cdot \epsilon_2 \text{ iff } s \cdot m < t \cdot n, \text{ or} \\ s \cdot m = t \cdot n \text{ and } \epsilon_1 < \epsilon_2 \text{ w.r.t. the order on } \mathcal{E}_i(e), \\ \text{with } m, n \text{ multiples of } e \in \mathcal{E}_{i-1}. \square \end{array}$$

EXAMPLE 3.9. Let K be a field of any characteristic and let the polynomial ring $S = K[x_0, \ldots, x_5]$ be endowed by the term ordering DegRevLex. Consider the graded module $M = S/I$, where the monomial ideal I is the face ideal of a triangulation of the real projective plane (first introduced by Reisner (1976)).

$$\begin{array}{rl} I = \langle & x_2x_4x_5,\ x_0x_4x_5,\ x_2x_3x_5,\ x_1x_3x_5, x_0x_1x_5, \\ & x_1x_3x_4,\ x_0x_3x_4,\ x_1x_2x_4,\ x_0x_2x_3, x_0x_1x_2\ \rangle. \end{array}$$

Then, a Schreyer frame Ξ for M is given by the bases:

$$\begin{array}{rl} B_1 = \{ & x_2x_4x_5,\ x_0x_4x_5,\ x_2x_3x_5,\ x_1x_3x_5,\ x_0x_1x_5,\ x_1x_3x_4,\ x_0x_3x_4,\ x_1x_2x_4, \\ & x_0x_2x_3,\ x_0x_1x_2\ \}; \end{array}$$

$$\begin{array}{rl} B_2 = \{ & x_4e_3,\ x_5e_6,\ x_5e_7,\ x_5e_8,\ x_2e_2,\ x_4e_5,\ x_2e_4,\ x_5e_9,\ x_3e_5,\ x_5e_{10},\ x_3e_8, \\ & x_4e_9,\ x_1e_7,\ x_4e_{10},\ x_3e_{10},\ x_0x_4e_4\ \}; \end{array}$$

$$B_3 = \{\ x_5e_{11},\ x_5e_{12},\ x_4e_9,\ x_5e_{13},\ x_5e_{14},\ x_5e_{15},\ x_4e_{15},\ x_2e_{16}\ \};$$

$$B_4 = \{\ x_5e_7\ \}.$$

For simplifying the notation, we have used the same letters e_j for indicating the elements of the canonical bases of the free modules $F_1 = S^{10}, F_2 = S^{16}, F_3 = S^8$. The term ordering on Ξ is defined for any level i in the usual way:

$$\begin{array}{l} s \cdot e_j < t \cdot e_k \text{ iff } s \cdot m_j < t \cdot m_k, \text{ or} \\ s \cdot m_j = t \cdot m_k \text{ and } j < k, \end{array}$$

where s, t are power products, e_j, e_k are elements of $\mathcal{E}_i$ s.t. $\xi_i(e_j) = m_j, \xi_i(e_k) = m_k$.

4. The Algorithms

An algorithm for computing Schreyer resolutions has been found independently by Richman (1974); Spear (1977); Schreyer (1980). In this section we propose a new version of that algorithm which improves both the computation of the Schreyer resolution, and its minimalization (in the graded case). In the graded case, minimalization can be a time-consuming process. The algorithm presented here allows a nice, efficient algorithm for computing a minimal resolution, and the graded Betti numbers of the minimal resolution are produced with no extra work.

Let $M = F_0/I$ be an R-module, let $\bar{\mathcal{C}}_1$ be an irredundant Gröbner basis of I, and let Ξ be a Schreyer frame of M. The following algorithm takes these data and produces a Schreyer resolution Φ of M, such that $\mathrm{in}(\Phi) = \Xi$. Of course we assume these sequences are truncated when they are infinite. The main idea here is very simple: process the s-pairs in such a way that the higher-level elements are processed first. If an element reduces to a non-zero value (which is possible since we do not yet have all of the elements of the Gröbner basis), then we get two elements for the price of one: we get the syzygy, and we get a Gröbner basis element at the previous level. We show that the Betti numbers are a by-product of this computation and we can minimalize Φ very easily. Note that we could modify this algorithm to compute the Gröbner basis on the fly. The difficulty is that we need then to compute the Schreyer frame on the fly as well, and the corresponding algorithm is somewhat harder to describe.

For this algorithm, let the variable $\mathcal{B}$ be the union of the sets $\mathcal{B}_1, \ldots, \mathcal{B}_l$ initialized as the monomial bases of Ξ. We call the elements of $\mathcal{B}$ the *S-polynomials*. The output consists of the Gröbner bases $\mathcal{C}_1, \ldots, \mathcal{C}_l$ of Φ together with the sets of minimal syzygies $\mathcal{H}_1, \ldots, \mathcal{H}_l,\ \mathcal{H}_i \subset \mathcal{C}_i$.

ALGORITHM 4.1. `Resolution[`$\bar{\mathcal{C}}_1$`]`

$\mathcal{C}_i, \mathcal{H}_i$`:=` $\emptyset$ $(1 \le i \le l)$
`while` $\mathcal{B} \neq \emptyset$ `do`
  m`:= min`$\mathcal{B}$
  $\mathcal{B}$`:=` $\mathcal{B} \setminus \{m\}$
  i`:=` $\mathrm{lev}(m)$
  `if` $i = 1$ `then`
    g`:= the element of` $\bar{\mathcal{C}}_1$ `s.t.` $\mathrm{lm}(g) = m$
    $\mathcal{C}_1$`:=` $\mathcal{C}_1 \cup \{g\}$
    $\mathcal{H}_1$`:=` $\mathcal{H}_1 \cup \{g\}$
  `else`
    (f, g)`:= Reduce[`$m, \mathcal{C}_{i-1}$`]`
    $\mathcal{C}_i$`:=` $\mathcal{C}_i \cup \{g\}$
    `if` $f \neq 0$ `then`
      $\mathcal{C}_{i-1}$`:=` $\mathcal{C}_{i-1} \cup \{f\}$
      $\mathcal{B}$`:=` $\mathcal{B} \setminus \{\mathrm{lm}(f)\}$ *removing one S-polynomial for free!!*
    `else`
      $\mathcal{H}_i$`:=` $\mathcal{H}_i \cup \{g\}$
`return` $\mathcal{C}_i, \mathcal{H}_i$ $(1 \le i \le l)$

The function `min` concerns the selection strategies we use for the algorithm. By definition, a *strategy* of `Resolution` is any total ordering of the set $\mathcal{B} = \mathcal{B}_1 \cup \ldots \cup \mathcal{B}_l$ s.t. if one of the following conditions holds:

(i) $\deg(m) - \mathrm{lev}(m) \le \deg(n) - \mathrm{lev}(n)$ and $\mathrm{lev}(m) < \mathrm{lev}(n)$;
(ii) $\deg(m) < \deg(n)$ and $\mathrm{lev}(m) = \mathrm{lev}(n)$;
(iii) $\deg(m) = \deg(n)$ and $\mathrm{lev}(m) > \mathrm{lev}(n)$;

then $m < n$, for all $m, n \in \mathcal{B}$. Note that such strategies are easily defined. For instance, a class of them is obtained by putting:

$$m < n \text{ iff } \deg(m) < \deg(n), \text{ or } \deg(m) = \deg(n) \text{ and } \mathrm{lev}(m) > \mathrm{lev}(n).$$

We call these strategies *DRLv* (Degree Reverse Level). Another natural class of strategies is *SDLv* (Slanted Degree Level):

$$\begin{aligned} &m < n \text{ iff } \deg(m) - \mathrm{lev}(m) < \deg(n) - \mathrm{lev}(n), \text{ or} \\ &\deg(m) - \mathrm{lev}(m) = \deg(n) - \mathrm{lev}(n) \text{ and } \mathrm{lev}(m) < \mathrm{lev}(n). \end{aligned}$$

The function `Reduce` in the algorithm `Resolution` is a simplification procedure which returns both the reductum and the corresponding syzygy. We may either reduce until the lead term cannot be simplified, or we can reduce until all terms cannot be simplified. (When we discuss implementations, the first variant will be algorithm A0, and the second variant, A1):

PROCEDURE 4.2. `Reduce`$[t \cdot \epsilon, \mathcal{C}_{i-1}]$

f := $t \cdot k$, where $\varphi_{i-1}(\epsilon) = k$
g := $t \cdot \epsilon$
while $f \neq 0$ and $\mathrm{lm}(f) \in \mathrm{in}\langle \mathcal{C}_{i-1} \rangle$ do
  choose $h \in \mathcal{C}_{i-1}$ s.t. $\mathrm{lm}(h) \mid \mathrm{lm}(f)$ (at first iteration h not allowed to be f)
  f := $f - \frac{\mathrm{lc}(f)\,\mathrm{lpp}(f)}{\mathrm{lc}(h)\,\mathrm{lpp}(h)}\, h$
  g := $g - \frac{\mathrm{lc}(f)\,\mathrm{lpp}(f)}{\mathrm{lc}(h)\,\mathrm{lpp}(h)}\, e$, where $\varphi_{i-1}(e) = h$
if $f \neq 0$ then
  g := $g - e$, where $\varphi_{i-1}(e) = f$
return f, g

PROCEDURE 4.3. `ReduceAll`$[t \cdot \epsilon, \mathcal{C}_{i-1}]$

f := $t \cdot k$, where $\varphi_{i-1}(\epsilon) = k$
g := $t \cdot \epsilon$
r := 0
while $f \neq 0$ do
  if $\mathrm{lm}(f) \in \mathrm{in}\langle \mathcal{C}_{i-1} \rangle$ then
    choose $h \in \mathcal{C}_{i-1}$ s.t. $\mathrm{lm}(h) \mid \mathrm{lm}(f)$ (at first iteration h not allowed to be f)
    f := $f - \frac{\mathrm{lc}(f)\,\mathrm{lpp}(f)}{\mathrm{lc}(h)\,\mathrm{lpp}(h)}\, h$
    g := $g - \frac{\mathrm{lc}(f)\,\mathrm{lpp}(f)}{\mathrm{lc}(h)\,\mathrm{lpp}(h)}\, e$, where $\varphi_{i-1}(e) = h$
  else
    r := $r + \mathrm{lm}(f)$
    f := $f - \mathrm{lm}(f)$
if $r \neq 0$ then
  g := $g - e$, where $\varphi_{i-1}(e) = r$
return r, g

To prove the correctness of the proposed algorithm, we start with the following:

PROPOSITION 4.4. *The resolution* Φ *computed by* `Resolution` *is a Schreyer one.*

PROOF. By induction on i, we suppose that at termination the set C_i is an irredundant Gröbner basis of the module $\ker \varphi_{i-1}$. By the definition of `Resolution` we have immediately that $\mathcal{B}_{i+1} \subset \mathrm{in}(\mathcal{C}_{i+1})$ i.e. $\mathcal{C}_{i+1}$ is a Gröbner basis of $\ker \varphi_i$. We have to prove now that $\mathcal{C}_{i+1}$ is an irredundant Gröbner basis, that is:

$$B_{i+1} = \mathrm{in}(\mathcal{C}_{i+1}).$$

Let $f \in \mathcal{C}_{i+1}$ be an element obtained at some step of the algorithm `Resolution` by a monomial $m \in \mathcal{B}$. We claim that $\mathrm{lm}(f) \in \mathcal{B}_{i+1}$. If $m \in \mathcal{B}_{i+1}$ the claim follows immediately since $\mathrm{lm}(f) = m$. Suppose now that $m \in \mathcal{B}_{i+2}$, i.e. f is the reductum of an S-polynomial. Denote by g any element of $\mathcal{C}_{i+1}$ computed at a previous step. By definition of the reductum we have that $\mathrm{lm}(g)$ does not divide $\mathrm{lm}(f)$. Moreover, by induction we can suppose that $\mathrm{lm}(g) \in \mathcal{B}_{i+1}$. Therefore, $\mathrm{lm}(g)$ is not a proper multiple of $\mathrm{lm}(f)$ since $\mathcal{B}_{i+1}$ is the minimal monomial basis of $\mathrm{in}(\ker \varphi_i)$. We conclude that the claim is true.

From this point on we assume that the module M is *graded*. Note that for any level i, the elements of $\mathcal{C}_i \setminus \mathcal{H}_i$ are non-minimal elements of the basis $\mathcal{C}_i$. By eliminating them, a new graded free resolution of M:

$$\Psi : \cdots \to G_q \xrightarrow{\psi_q} G_{q-1} \xrightarrow{\psi_{q-1}} \cdots \xrightarrow{\psi_2} G_1 \xrightarrow{\psi_1} F_0 \to M \to 0$$

is clearly defined with $\mathrm{rk}(G_i) \leq \mathrm{rk}(F_i)$, where possibly $G_i = 0$ even if $F_i \neq 0$. Denote by $\mathcal{D}_i$ the image of the basis of free module G_i through the map ψ_i $(i \geq 1)$. We have a graded epimorphism of complexes $\Phi \xrightarrow{\theta} \Psi$:

$$\begin{array}{ccccccccccc}
\ldots & \to & F_q & \to & \ldots & \to & F_1 & \to & F_0 & \to & M \\
 & & \theta_q \downarrow & & & & \theta_1 \downarrow & & \downarrow & & \\
\ldots & \to & G_q & \to & \ldots & \to & G_1 & \to & F_0 & \to & M
\end{array}$$

such that $\theta_{i-1}(\mathcal{H}_i) = \mathcal{D}_i$ and $\mathcal{H}_1 = \mathcal{D}_1$. For $i > 1$, denote by $\mathcal{K}_i$ the subset of $\mathcal{C}_i$ of the syzygies traced by S-polynomials which reduce to non-zero elements. Let g be an element of $\mathcal{K}_i$ obtained at any step of the procedure `Resolution` and denote by f the corresponding S-polynomial reductum. Note that the component corresponding to f is zero for the elements of $\mathcal{K}_i$ computed in the previous steps, and is equal to $-1 \in K$ for the syzygy g. Therefore, $\mathcal{K}_i$ is a free basis of the kernel of the projection defined by θ_{i-1} of the module $\ker(\varphi_{i-1})$ onto $\ker(\psi_{i-1})$.

For the algorithm `Resolution` we have the following important result:

PROPOSITION 4.5. *The resolution* Ψ *is a minimal resolution of the graded module* M*.*

PROOF. By the conditions (ii) and (iii), any element f of the basis $\bar{C}_1$ is added by `Resolution` to $\mathcal{H}_1 = \mathcal{D}_1$ only if f is not reducible to zero w.r.t. to the partial Gröbner basis of the elements of degree $\leq \deg(f)$. Therefore, $\mathcal{D}_1$ is a minimal basis of the module $I = image(\varphi_1)$.

We have to prove now that $\mathcal{D}_2, \ldots, \mathcal{D}_q$ define a minimal resolution of $\ker \psi_1$. Initially, we show that any element of $\mathcal{H}_i$ is not a linear combination over the ring S of other elements of $\mathcal{C}_i$. Then, we prove that $\mathcal{D}_i = \theta_{i-1}(\mathcal{H}_i)$ is a minimal basis of the module $\ker \psi_{i-1}$.

Let $f \in \mathcal{C}_i, \deg(f) = d$. Suppose that f is not a minimal element of the basis $\mathcal{C}_i$. Then, there is $g \in \mathcal{C}_{i+1}$ a syzygy of degree d s.t. its component corresponding to f is different from zero. We claim that f, g are obtained in the algorithm `Resolution` by the same monomial of $\mathcal{B}_{i+1}$.

Let m, n be the monomials of the Schreyer frame which create f, g respectively. Note that m, n have the same degree d, but $\mathrm{lev}(m) \leq \mathrm{lev}(n)$. Since the syzygy g has a non-zero component corresponding to f, we have clearly that the monomial m cannot be selected in the algorithm *after* n. On the other hand, by the strategy condition (iii) of `Resolution`, we have that m cannot be selected *before* n. Therefore, the elements f, g are created *simultaneously* by a monomial $m = n$ of the level $i + 1$. We deduce that any element of $\mathcal{H}_i$ is minimal in the basis $\mathcal{C}_i$.

By contradiction, we suppose now that $\mathcal{D}_i$ is not a minimal basis of the module $\ker \psi_{i-1}$, i.e. there exist $f' \in \mathcal{D}_i, g' \in \ker \psi_i$ s.t. $\deg(f') = \deg(g')$ and g' has the component corresponding to f' different from zero. Therefore, by definition of θ, there are $f \in \mathcal{H}_i, g \in \ker \varphi_i$ of the same degree s.t. g has a non- zero component corresponding to f. This contradicts the minimality of the elements of $\mathcal{H}_i$.□

Note that from the above result it follows that the graded Betti numbers of M are obtained by the algorithm `Resolution` with no additional computations. They are simply the number of elements of the sets $\mathcal{H}_i$ in each degree. Moreover, for obtaining a minimal resolution of the module M it is sufficient to transform each $\mathcal{H}_i,\ i > 1$ in $\mathcal{D}_i$ by means of the homomorphism θ_{i-1}. The following simple algorithm implements θ_{i-1}.

Denote by $\hat{\mathcal{K}}_i$ the subsets of $\mathcal{B}_i$ of the monomials corresponding to S-polynomials which reduce to non-zero elements in `Resolution`. Moreover, assume that $\hat{\mathcal{K}}_i$ is endowed by the converse ordering w.r.t. the selection strategy.

ALGORITHM 4.6. `Minimalize`$[\mathcal{H}_i]$

```
for m ∈ K̂_i do
  (f, g):= resp. the reductum and the syzygy obtained by m
  for h ∈ H_i do
    h_e:= the component of h corresponding to e ∈ E_{i-1}, φ_{i-1}(e) = f
    h:= h + h_e · g
    h:= Strip[h]
return H_i
```

The subprocedure `Strip` is defined as follows:

PROCEDURE 4.7. `Strip`$[h]$

```
for m ∈ K̂_{i-1} do
  g:= the syzygy of C_{i-1} obtained by m
  h_e:= the component of h corresponding to e ∈ E_{i-1}, φ_{i-1}(e) = g
  h:= h − h_e · e
return h
```

The correctness of `Minimalize` is based on the remark that if the syzygy $g \in \mathcal{C}_i$ is com-

puted at any step of the procedure `Resolution`, then the components of g corresponding to the elements of $\mathcal{C}_{i-1}$ obtained in the next steps are clearly all zero. Note that our minimalization of the Schreyer resolution is much easier than the "classical" one since we know *a priori* which syzygies give rise to a minimal resolution.

EXAMPLE 4.8. Applying the algorithm `Resolution` for $\mathrm{char}(K) \neq 2$ to the module $M = S/J$ of Example 3.9, we get the following Schreyer resolution:

$$\begin{aligned}
C_1 = \{\ & x_2x_4x_5,\ x_0x_4x_5,\ x_2x_3x_5,\ x_1x_3x_5,\ x_0x_1x_5,\ x_1x_3x_4,\ x_0x_3x_4,\ x_1x_2x_4,\\
& x_0x_2x_3,\ x_0x_1x_2\ \};
\end{aligned}$$

$$\begin{aligned}
C_2 = \{\ & x_4e_3 - x_3e_1,\ x_5e_6 - x_4e_4,\ x_5e_7 - x_3e_2,\ x_5e_8 - x_1e_1,\ x_2e_2 - x_0e_1,\\
& x_4e_5 - x_1e_2,\ x_2e_4 - x_1e_3,\ x_5e_9 - x_0e_3,\ x_3e_5 - x_0e_4,\ x_5e_{10} - x_2e_5,\\
& x_3e_8 - x_2e_6,\ x_4e_9 - x_2e_7,\ x_1e_7 - x_0e_6,\ x_4e_{10} - x_0e_8,\ x_3e_{10} - x_1e_9,\\
& x_0x_4e_4 - x_1x_3e_2\ \};
\end{aligned}$$

$$\begin{aligned}
C_3 = \{\ & x_5e_{11} - x_3e_4 + x_2e_2 + x_4e_7 + x_1e_1,\\
& x_5e_{12} - x_4e_8 + x_2e_3 - x_0e_1 + x_3e_5,\\
& x_4e_9 - x_3e_6 - e_{16},\\
& x_5e_{13} - x_1e_3 + x_0e_2 - e_{16},\\
& x_5e_{14} - x_4e_{10} + x_0e_4 - x_2e_6 - x_1e_5,\\
& x_5e_{15} - x_3e_{10} + x_1e_8 - x_2e_9 - x_0e_7,\\
& x_4e_{15} - x_3e_{14} + x_1e_{12} - x_0e_{11} + x_2e_{13},\\
& 2 \cdot (x_2e_{16} + x_0x_4e_7 + x_0x_1e_1 - x_1x_3e_5)\ \};
\end{aligned}$$

$$C_4 = \{\ x_5e_7 - x_4e_6 + x_3e_5 - x_1e_2 + x_0e_1 - x_2e_4 - x_2e_3 - e_8\ \}.$$

During the computation two monomials of the Schreyer frame:

$$\begin{aligned}
& x_4e_9 \in \mathcal{B}_2,\\
& x_5e_7 \in \mathcal{B}_3
\end{aligned}$$

give rise to non-zero S-polynomial reductions. The couples of non-minimal elements of the resolution obtained by these monomials are respectively:

$$\begin{aligned}
& x_0x_4e_4 - x_1x_3e_2 \in \mathcal{C}_1,\\
& x_4e_9 - x_3e_6 - e_{16} \in \mathcal{C}_2;
\end{aligned}$$

$$\begin{aligned}
& 2 \cdot (x_2e_{16} + x_0x_4e_7 + x_0x_1e_1 - x_1x_3e_5) \in \mathcal{C}_2,\\
& x_5e_7 - x_4e_6 + x_3e_5 - x_1e_2 + x_0e_1 - x_2e_4 - x_2e_3 - e_8 \in \mathcal{C}_3.
\end{aligned}$$

By removing them by means of the algorithm `Minimalize`, we get a minimal resolution Ψ of the graded module M. Clearly something changes if the characteristic of the base field K is 2. In that case the monomials:

$$\begin{aligned}
& x_2e_{16} \in \mathcal{B}_2,\\
& x_5e_7 \in \mathcal{B}_3
\end{aligned}$$

hold reductions to zero, i.e. minimal syzygies.

5. Implementation

In this section we examine some of the choices that need to be made by an implementation and propose some optimizations. After discussing these choices and implementation issues, we describe four variants of our algorithm: A0, A1, A1-H, and B, as well as two other algorithms, M and MH. In the next section, we give examples of resolution computations using these algorithms and variants, to evaluate their relative usefulness, at least for these examples.

With the notation of Section 4, let $M = F_0/I$ be an R-module, and let

$$\Xi : F_l \xrightarrow{\xi_l} F_{l-1} \xrightarrow{\xi_{l-1}} \cdots \xrightarrow{\xi_2} F_1 \xrightarrow{\xi_1} F_0$$

be a Schreyer frame of the module M. For all i, we have defined:

$\mathcal{E}_i$ = the canonical basis of F_i;
$\mathcal{E}_i(e) = \{\epsilon \in \mathcal{E}_i \ : \ \xi_i(\epsilon) = s \cdot e, \text{ for some power product } s\}$;
$\mathcal{B}_i = \xi_i(\mathcal{E}_i)$, the S-polynomials to reduce in level i;

where e is any element in $\mathcal{E}_{i-1}$.

It is also useful to define, for $\epsilon \in \mathcal{E}_i$, the *total monomial* and *total power product* of ϵ:

If $\epsilon \in \mathcal{E}_0$, set $\text{total}(\epsilon) := \epsilon$ and $\text{totalpp}(\epsilon) := 1$. If $i \geq 1$, and $\epsilon \in \mathcal{E}_i$, set $\text{total}(\epsilon) := s \cdot \text{total}(e)$ and $\text{totalpp}(\epsilon) := s \cdot \text{totalpp}(e)$, where $\xi_i(\epsilon) = s \cdot e$.

5.1. Implementation issues

We discuss the main issues that an implementation must address.

Compute the Gröbner basis first, or on the fly?

If the input module M is not graded, then a Gröbner basis must be computed first. If M is graded, then one can either precompute the Gröbner basis of I, or compute it during the computation of the resolution. The advantage to computing it along with the resolution is that one does not duplicate any effort this way. The advantages of precomputing the Gröbner basis include: the time required is small compared to the computation of the entire resolution; the algorithm is much easier to describe and implement; and certain other optimizations are made possible by having the entire frame available.

Algorithms A0, A1, and A1-H all precompute the Gröbner basis, while Algorithm B computes it on the fly.

The choice of the induced term orders

Induced term orders is the single most important optimization to use for computing resolutions. For the algorithms presented here, it is necessary for the correctness of the algorithm. For other algorithms (e.g. M and MH presented below), it is essential for obtaining a reasonable performance from the algorithm.

However, there is a lot of leeway in choosing an induced term order.

This choice determines the Schreyer frame, and so it is important to choose the order so that the frame is as small as possible. During the algorithm for construction of the frame, the frame (and therefore the term order) is determined by the choice of total orders on

Table 1.

L+	Increasing lexicographic order;
L-	Decreasing lexicographic order;
DL+	Increasing degree, and in each degree, increasing lex order;
DL-	Increasing degree, and in each degree, decreasing lex order;
RL+	Increasing lex order w.r.t. the reversed set of variables;
RL-	Decreasing lex order w.r.t. the reversed set of variables;
DRL+	Increasing degree, and in each degree, increasing lex order w.r.t. the reversed set of variables;
DRL-	Increasing degree, and in each degree, decreasing lex order w.r.t. the reversed set of variables;
O+	Ascending in the given term ordering (e.g. same as DRL- for deg-rev-lex order);
O-	Descending in the given term ordering.

the sets $\mathcal{E}_i(e)$, for each i and e (Lemma 3.5 and Proposition 3.8). For each i and e, if $\{\epsilon_1, \ldots, \epsilon_r\} = \mathcal{E}_i(e)$, then we need to choose a total order on the set of monomials of S, $\{t_1, \ldots, t_r\}$, where $\xi(\epsilon_i) = t_i \cdot e$. Note that these need not be term orders, and in fact, could be arbitrarily chosen. Experience indicates that this would be a bad idea. Instead, the orders that we consider for the $\{t_1, \ldots, t_r\}$ are summarized in Table 1.

Some of these are quite bad (O- for one) and are included only for comparison purposes. Overall, RL- and DRL+ often outperform the other possible orders. Algorithms A0, A1, A1-H presented below allow any one of these to be used for construction of the Schreyer frame (and therefore also the term order). Algorithm B uses a somewhat different order, described below (essentially DL-, if the monomial order on the ring S is the degree reverse lexicographic order).

The order to process S-polynomials

In the algorithm `Resolution` the choice for the selection strategy is quite free, except for some conditions which provide the minimality of the syzygies obtained by reductions to zero. Among the classes of orderings compatible with these conditions we choose to use SDLv (see Section 4), (slanted degree level by level), which is well-suited for partial computations. It appears also that SDLv provides a better partial auto-reduction of the resolution than DRLv.

In each (degree,level), the order to perform the reductions is irrelevant to the correctness of the algorithm. However, since a reduction to a non-zero element determines a Gröbner basis element at the previous level, the choice may affect the sizes of these elements, and consequently the running time of the algorithm.

Our experience indicates that using the same order as that for the $\mathcal{E}_i(e)$ is a good choice. More precisely, for two elements $e_1, e_2 \in \mathcal{E}_i$ of the same degree, process e_1 before e_2 if $\text{totalpp}(e_1) < \text{totalpp}(e_2)$, where this order is one of the orders above (L+, L-, etc). The tiebreaker order (if $\text{totalpp}(e) = \text{totalpp}(f)$) does not seem to be very important.

For our algorithms A0, A1, A1-H below, we process the S-polynomials using the same order as that used to determine the induced term order. This is not necessary, but our experience is that it is a good choice.

The amount of auto-reduction

While performing a single reduction, we may use the `Reduce`, or the `ReduceAll` algorithm. The first stops once the lead term cannot be simplified, and the latter stops once no term can be simplified. This second approach often leads to partially "auto-reduced" Gröbner bases. One could also continue, making each Gröbner basis totally auto-reduced. In some examples, this gives a dramatic improvement, since the size of the basis will often be much smaller. The difficulties include the fact that minimalization is conceivably much more time consuming, as is modifying the elements so far obtained.

We have not yet implemented minimalization for this full auto- reduction, but initial time tests indicate that full auto-reduction would be quite useful in certain cases. We will report on these techniques in a later paper.

Algorithms A0 and B use no auto-reduction (i.e. `Reduce`), while algorithms A1 and A1-H use `ReduceAll`.

Reduction strategy

While performing a single reduction, if there are several possible divisors of the lead term, which one should we choose? Our choice (in Algorithms A0,A1,A1-H) is to choose the element which is least, in the monomial order. This is necessary if partial auto-reduction is desired. In general, it is often the case that only one element of the basis divides the given monomial (especially further back in the resolution). It would be nice to use this fact in some optimization.

Representation of monomials and the induced monomial order

Given a monomial $m = t \cdot e \in F_i$, how should we represent m, in order to allow for the fast computation of the induced monomial order? We have chosen to represent m as the pair (totalpp(m), e), instead of using the pair (t, e). This complicates general arithmetic, but speeds up the comparison times. We have not done time comparisons between the two approaches, but we expect that the added complications of using the first technique leads to better performance of the algorithm.

Consider a heap based approach for performing reductions

If we are reducing a very large polynomial, and in each iteration, we add in small polynomials, then we are essentially performing an insertion sort. If we use an analog of a heap sort, reduction times are often drastically reduced. See Yan (1998) for details. Our algorithm A1-H uses this approach.

Allow for partial computation

This is just a reminder to implementors that allowing for partial computation (e.g. computing to a certain (slanted) degree, or a certain level) is essential, since one often only cares about part of the resolution, e.g. the (slanted) linear part, or, in the case $R = S/I$, because the entire resolution often cannot be computed since the resolution is infinite.

Miscellaneous optimizations

Before performing the reductions of the S-polynomials, one may compute the resolution of the monomial module $F_0/\operatorname{in}(I)$. The regularity of F_0/I is no larger than the regularity of this monomial module. Therefore one may avoid processing all S-polynomials in high enough (slanted) degree. Furthermore, the computation of the resolution of $F_0/\operatorname{in}(I)$ involves very little extra work.

Another optimization is that, since we compute a Gröbner basis anyway, it is quite simple to check whether we have a complete intersection. If so, simply return the Koszul complex (or, in the module case, the Eagon–Northcott complex) of the original generators. These optimizations are not used in the algorithms A0, A1, A1-H, B, M, MH, since this decision rightly belongs in a higher level routine.

Finally, the Schreyer frame will sometimes be longer than the minimal resolution. One can postpone computing certain (degree,level)'s (possibly forever, if we are only computing up to a specific slanted degree). See Example 6.4 for a specific case of this. This speeds up some partial computations quite a lot. (Our algorithms A0, A1, A1-H use this optimization.)

5.2. Algorithms

We now summarize the algorithms and variants that we test in the next section.

- Algorithm A0. This is the algorithm `Resolution` described in this paper. First we compute a Gröbner basis of I, and then its Schreyer frame. We then process the elements in any way s.t. in a given degree, we process elements of higher level first. For example, processing elements by increasing slanted degree is the method we have implemented in Macaulay2. In this algorithm, the order in which pairs are reduced in each (degree, level) is important. Heuristically, we have found that using the same order as that used to compute the frame (up to computing degree by degree) is best most of the time. The reduction of an element stops once a lead term is obtained where the corresponding Gröbner basis element at the previous level has not been computed. This tends to not give auto-reduced Gröbner bases, but is very effective in many examples.
- Algorithm A1. This is a variant of algorithm A0, where we use the simplification procedure `ReduceAll`.
- Algorithm A1-H. We have also implemented a *heap based* reduction algorithm for algorithm A1. See Yan (1998) for a description of the technique (the author calls them "geobuckets"). This often improves performance by a modest amount, although the improvement for computing Gröbner bases in general can be dramatic.
- Algorithm B. This is essentially algorithm A0, with the following differences. The Gröbner basis of I is computed "on the fly", as is the Schreyer frame. This means that the Gröbner basis computation of I is not duplicated. On the down side, it means that the ordering we use to construct the frame must be essentially degree by degree based, which we have found not to be optimal in many cases. Also, the coding of this algorithm is sufficiently more complicated that it is much harder to change the algorithm for testing purposes. Thus, we cannot change the frame order, we do not have heap based reduction, and we do not have the `ReduceAll` routine implemented. If we did, one could expect corresponding increases in performance.

The term ordering used to compute the frame, as well as process elements, is DO+ that is increasing degree, and in each degree, ascending in the given term ordering.

- ALGORITHM M. Compute a minimal generating set of syzygies on the minimal generating set of the input module I, and then compute a minimal generating set of syzygies of those elements, and continue until done. At each step, use an induced term ordering. Also, compute minimal generators while computing the syzygies, to not duplicate effort.

  This was the method used in the original Macaulay program, except that we did not use induced orders for the term ordering. The savings obtained from using induced orders is dramatic.
- ALGORITHM MH. Same as M, except compute Hilbert functions so that at each level, degree, one knows exactly the sum of the numbers of Gröbner basis elements and minimal syzygies at that step. The Hilbert functions take some time to compute, but the savings can be dramatic. Both algorithms M and MH are described in more detail in Capani *et al.* (1997).

6. Examples and Timings

For each example, we describe the ideal $I \subset R$, and give the minimal Betti numbers and a table of statistics for computing the resolution of R/I. In the case that random polynomials are required, the exact same ideal is used for each algorithm variant.

For each example, the table of statistics has the following information. The column **Frame** represents the size of the Schreyer frame in terms of total number of monomials w.r.t. the different choices of the ordering. Each **Res** is the size of the Schreyer resolution (total number of monomials in all the Gröbner bases) for the algorithms corresponding to the previous columns. For any algorithm we give the computing time w.r.t. to the different frame orderings.

All the times are reported in seconds. The tests are performed with a NEC Versa 6200MX laptop, running Linux 2.0.29, 128 MB RAM, and a 166 MHz Pentium processor. Times run on a Pentium Pro 200 Mhz Linux box tend to run roughly 2.5 to 3 times faster than on this Pentium processor. The version of Macaulay2 that we use is 0.8.30. All of the examples and benchmarks used here are included in the Macaulay2 distribution.

EXAMPLE 6.1. [3×3 COMMUTING MATRICES] $I \subset R = \mathbf{Z}_{32003}[x_{i,j}, y_{i,j}, 1 \leq i, j \leq 3]$ is the ideal generated by the entries of the 3×3 matrix $XY - YX$, where $X = (x_{i,j})$ and $Y = (y_{i,j})$. This is an ideal minimally generated by eight quadrics in 18 variables.

Although this is a simple example, we have included it since it is one of our favorites, and it illustrates some of the pitfalls in implementing resolution algorithms.

Note that, even in this simple case, there are orders for which the algorithm has extremely poor performance. Using the orders O- or L- for computing the Schreyer frame leads to enormous frames, and consequently to very large resolutions. From now one, we will not include either of these two orders in our tests.

Note also that the partially auto-reduced resolution for each choice of order is larger than the size of the resolution when computed using Algorithm A0. The timings are only accurate up to about 10% or 15% (the previous time that the algorithm MH was run, the time required was 1.1 s).

This is an example where full auto-reduction gives larger resolutions, in terms of numbers of monomials. Almost all of the current algorithms now compute this quite rapidly.

Order	Frame	A0	Res	A1	A1-H	Res
L+	674	0.99	20658	1.13	1.01	23283
RL+	742	1.1	22753	1.68	1.44	25475
DL+	682	0.99	20755	1.14	1.	23320
DL-	774	1.4	24175	1.69	1.5	29275
DRL+	742	1.1	22790	1.66	1.41	25112
DRL-	650	0.97	19214	1.13	0.97	20945
O+	650	0.97	19214	1.09	0.96	20945
RL-	1208	1.71	38762	3.29	2.7	72571
L-	1722	1.93	66493	10.72	6.48	213865
O-	2806	2.21	93842	59.19	20.67	578646
Algorithm B		time	1.3			
Algorithm M		time	1.2			
Algorithm MH		time	0.92			

The minimal Betti numbers of R/I:

```
Total  1  8  33  60  61  32  5
    0: 1  -  -   -   -   -  -
    1: -  8  2   -   -   -  -
    2: -  -  31  32  3   -  -
    3: -  -  -   28  58  32 4
    4: -  -  -   -   -   -  1
```

EXAMPLE 6.2. [GR(2,7)] $I \subset R = \mathbf{Z}_{31991}[x_1, \ldots, x_{21}]$ is the ideal of the grassmannian of 2-planes in affine 7-space, in its Plücker embedding. I is minimally generated by 35 quadrics. I is also generated by the 4×4 Pfaffians of a generic 7×7 skew symmetric matrix.

Order	Frame	A0	Res	A1	A1-H	Res
L+	9402	32.47	344575	26.44	22.22	284530
RL+	9814	37.44	390317	25.88	22.15	297173
DL+	9402	35.15	355338	29.16	23.34	303188
DL-	9814	38.04	411158	27.32	22.71	307689
DRL+	9814	40.96	411438	27.55	22.4	307689
DRL-	9402	36.02	355292	29.09	23.2	303188
O+	9402	33.83	355292	27.99	22.79	303188
RL-	9402	36.08	348923	25.87	21.32	278994
Algorithm B		time	34.6			
Algorithm M		time	471.33			
Algorithm MH		time	316.19			

All of the orders shown have comparable performance relative to each other. The best timings are obtained using auto-reduction, and heap based reduction. The times for algorithms M, MH are roughly a factor of 15–20 times worse. Heuristically, if the minimal Betti numbers are very small relative to the number of Gröbner basis elements at each level, then the algorithms M and MH will outperform A0, A1, A1-H, and B. Likewise, if the minimal Betti numbers are relatively large, the opposite is the case.

```
Total   1   35   140   385   819   1080   819   385   140   35   1
   0:   1   -    -     -     -     -      -     -     -     -    -
   1:   -   35   140   189   84    -      -     -     -     -    -
   2:   -   -    -     196   735   1080   735   196   -     -    -
   3:   -   -    -     -     -     -      84    189   140   35   -
   4:   -   -    -     -     -     -      -     -     -     -    1
```

EXAMPLE 6.3. [GR(3,6)] $I \subset R = \mathbf{Z}_{31991}[x_1, \ldots, x_{20}]$ is the ideal of the grassmannian of 3-planes in affine 6-space, in its Plücker embedding. I is minimally generated by 35 quadrics.

Order	Frame	A0	Res	A1	A1-H	Res
L+	7770	93.98	581327	28.98	23.47	269890
RL+	7770	70.78	467770	25.15	21.01	260080
DL+	7770	103.66	622576	35.13	27.98	301043
DL-	7770	81.55	524364	29.86	24.6	282836
DRL+	7770	81.53	524408	29.6	24.15	282836
DRL-	7770	104.23	622803	36.13	27.01	301043
O+	7770	102.69	622803	34.34	27.45	301043
RL-	7770	96.18	577457	24.62	21.1	256319
Algorithm B		time	80.25			
Algorithm M		time	*			
Algorithm MH		time	*			

This is the first algorithm that we are aware of that is able to compute this resolution, and in a reasonable time as well! Notice that the specific choice of order (of these eight orders, that is) does not affect the size of the Schreyer frame, and in fact, does not change the timings very much either.

What does make a difference is the use of partial auto-reduction (Algorithm A1). Notice that the heap based algorithm gives a modest performance boost to the A1 algorithm. Algorithms M, MH never finished on these examples (they ran out of memory, at about 128 MB).

It is very intriguing that $\mathrm{Gr}(3,6)$ and $\mathrm{Gr}(2,7)$ have the same Hilbert series. We would very much like to know an explanation of this fact.

The minimal Betti numbers:

```
Total   1   35   140   301   735   1080   735   301   140   35   1
    0:  1    -     -     -     -      -     -     -     -    -   -
    1:  -   35   140   189     -      -     -     -     -    -   -
    2:  -    -     -   112   735   1080   735   112     -    -   -
    3:  -    -     -     -     -      -     -   189   140   35   -
    4:  -    -     -     -     -      -     -     -     -    -   1
```

Example 6.4. [Linear section of tangent developable of rational normal curve of degree 13] For each integer $g \geq 6$, let $\tan(g)$ be the ideal in $R = \mathbf{Z}_{101}[x_1, \ldots, x_{g-3}]$ generated by the following quadrics:

$$(i + j - 1)\, x_{i-2}x_{j-2} - (ij)\, x_{i+j-3}x_{g-3}$$

for $2 \leq i \leq j \leq g - 2$, where $i + j \leq g - 1$, and the elements:

$$(2g - i - j - 1)\, x_{i-2}x_{j-2} - (g - i)(g - j)\, x_{i+j-g-1}x_{g-3}$$

where $2 \leq i \leq j \leq g - 2$, and $i + j > g - 1$.

These examples arise from the tangent developable of a rational normal curve of degree g by taking special hyperplane sections. In practice, we are only interested in the 2-linear strand of the resolution, as the rest of the graded Betti numbers can be deduced from these.

The timings given are for computing the 1-linear part of the resolution of $R/\tan(13)$.

Order	Frame	A0	Res	A1	A1-H	Res
DL-	20482	65.81	691230	69.59	72.84	678542
DRL+	20482	65.73	691230	70.2	73.62	678542
Algorithm B		time	95.52			
Algorithm M		time	554.62			
Algorithm MH		time	597.25			

The Betti numbers of the linear strand:

```
Total   1   55   320   891   1408   1155
    0:  1    -     -     -      -      -
    1:  -   55   320   891   1408   1155
```

The size of the frame in either case is given by:

```
Total   1   66   440   1485   3168   4620   4752   3465   1760   594   120   11
    0:  1    -     -      -      -      -      -      -      -     -     -    -
    1:  -   55   330    990   1848   2310   1980   1155    440    99    10    -
    2:  -   10   100    450   1200   2100   2520   2100   1200   450   100   10
    3:  -    1    10     45    120    210    252    210    120    45    10    1
```

After processing the 2310 S-polynomials at level 5 in slanted degree 1, obtaining 1155

minimal syzygies, the 1980 S-polynomials at level 6 in slanted degree 1 all give non-minimal syzygies. Thus the remaining S-polynomials in slanted degree 1 need never be reduced since we are only interested in the linear strand.

EXAMPLE 6.5. [GOR(8,3)] Given a homogeneous polynomial f of degree d in $R = \mathbf{Z}_{101}[x_1, \ldots, x_n]$, define the ideal:

$$I_f = \{g \in R : g(\partial/\partial x_1, \ldots, \partial/\partial x_n) \cdot f = 0\}.$$

For a random f of degree d, in R denote the ideal I_f by $\mathrm{gor}(n, d)$.

The timings given are for computing the graded Betti numbers of $R/\mathrm{gor}(8, 3)$.

Order	Frame	A0	Res	A1	A1-H	Res
L+	1794	407.94	382921	396.69	376.43	406677
RL+	1794	591.22	413960	592.14	544.9	418950
DL+	1794	684.32	475749	653.25	588.63	472475
DL-	1794	590.86	413960	589.8	502.71	418950
DRL+	1794	612.36	413960	596.46	571.62	418950
DRL-	1794	695.06	475749	660.24	599.89	472475
O+	1794	685.1	475749	666.18	578.55	472475
RL-	1794	396.66	382921	401.85	370.75	406677
Algorithm B		time	617.13			
Algorithm M		time	*			
Algorithm MH		time	*			

The issue here is auto-reduction. Full auto-reduction would be the best, compared to these other algorithms, but we do not yet have a complete implementation. Algorithms M, MH never finished on these examples.

The minimal Betti numbers:

```
Total  1  28  105  162  168  162  105  28  1
    0: 1  -   -    -    -    -    -    -   -
    1: -  28  105  162  84   -    -    -   -
    2: -  -   -    -    84   162  105  28  -
    3: -  -   -    -    -    -    -    -   1
```

EXAMPLE 6.6. [JAC(4,5)] Given a random homogeneous polynomial f of degree 4 in $R = \mathbf{Z}_{101}[x_1, \ldots, x_5]$, let $\mathrm{jac}(f)$ be the ideal in $R = R/f$ generated by the five partial derivatives of f.

The timings given are for computing the Betti numbers of $R/\mathrm{jac}(f)$ through F_4.

Order	Frame	A0	Res	A1	A1-H	Res
DL-	1032	1374.95	561913	1284.13	800.12	574837
DRL+	1032	1541.7	561913	1550.02	980.57	574837
Algorithm B		time	1325.8			
Algorithm M		time	9.03			

The Gröbner bases of the syzygy modules stabilize (in each case) to 203 elements each, while the minimal Betti numbers stabilize at 8 each. This is typically the kind of situation for which algorithms M, MH are much better. Such is the case here. Partial auto-reduction makes little difference, but the heap reduction algorithm A1-H gives better performance, but comes nowhere near the small amount of time that Algorithm M requires.

The minimal Betti numbers (up through level 4):

```
Total   1   4   7   8   8
    0:  1   -   -   -   -
    1:  -   -   -   -   -
    2:  -   -   -   -   -
    3:  -   4   1   -   -
    4:  -   -   -   -   -
    5:  -   -   -   -   -
    6:  -   -   6   4   1
    7:  -   -   -   -   -
    8:  -   -   -   -   -
    9:  -   -   -   4   6
   10:  -   -   -   -   -
   11:  -   -   -   -   -
   12:  -   -   -   -   1
```

6.1. SUMMARIES OF TIMINGS

Summaries of the timings of the above examples, and the Cocoa (Capani *et al.*, 1997) and Singular (Grassmann *et al.*, 1995) examples is given in Table 2. Some examples from these two sets are missing, since we were unable to duplicate the given ideals. Particularly in the Singular group of examples, it is possible that some of these ideals are given differently from those in Grassmann *et al.* (1995). For the exact examples that we have run, please check the Macaulay2 distribution. There is one exception to the reason for omission. The homogeneous cyclic 6 roots does not finish in reasonable time on the algorithms A0, A1, A1-H. We were also unable to duplicate similar running times to those in Capani *et al.* (1997) for the M, MH algorithms for the cocoa2 and the homogeneous cyclic 6 roots examples.

Most of the examples here have very high running times in Macaulay classic, since the monomial orders used there are not well suited for computing resolutions (they are not induced term orders). We have not included timings for the systems Singular and Cocoa, since we do not have access to their latest code. Hopefully the authors of these systems will publish their timings for all of these examples as well.

No one choice of orders gives the best performance in every case, but a good choice in almost all of these examples would be to use DRL+. For most of the examples of any complexity, using partial auto-reduction (Algorithm A1) is far superior to using Algorithm A0, and using the heap based reduction, A1-H is generally best. What appears to be true a large part of the time is that the smaller the frame, then the faster the algorithm, for each of algorithms A0 and A1.

Table 2.

Best	Example	MH	A0[DRL+]	A1-H[DRL+]	B
B	cocoa1	1.11	0.43	0.45	0.28
A1	cocoa2	922.42	19.63	7.37	22.14
A1,B,A0	cocoa3	1.81	1.02	0.98	0.99
MH	cocoa4	0.2	0.4	0.35	0.64
A1	cocoa5	5.56	3.34	2.96	3.09
MH,B	cocoa6	1.00	1.22	1.21	1.1
B	cocoa8	3.42	1.28	1.27	1.07
A1	gr27	316.19	40.96	22.71	34.6
A1	gr36	*	81.53	24.15	80.25
A1	gor83	*	612.36	571.62	617.13
A1	tandev13	597.25	65.73	73.62	95.52
MH	jac45	9.03	1541.7	980.57	1325.8
A0	singular3	2.71	1.16	1.4	0.5
B	singular4	19.74	8.74	9.8	4.84
B	singular5	5.34	1.21	1.8	0.96
MH	singular7	1.26	3.19	2.8	16.15
B	singular8	1.02	0.26	0.22	0.16
MH	singular9	16.12	39.68	33.35	38.3
MH	singular10	0.42	1.42	1.39	1.8
MH	singular11	44.66	245.02	245.69	281.93
A1	singular12	35.04	6.29	4.63	6.79
MH	singular13	5.87	13.53	13.41	3.92
A1	singular14	0.98	0.27	0.24	0.41
B,A01	singular19	3.2	0.64	0.65	0.61
B	singular20	126.77	101.3	98.52	94.15

Similarly, no one choice of algorithm always gives the best performance. However, we were able to obtain resolutions (e.g. $\mathrm{Gr}(3,6)$) that we have been unable to obtain using any other algorithm or theoretical result.

References

Capani, A., De Dominicis, G., Niesi, G., Robbiano, L. (1997). Computing minimal finite free resolutions, *J. Pure Appl. Algebra*, **117–118**, 105–117.

Grassmann, H., Greuel, G.M., Martin, B., Neumann, W., Pfister, G., Pohl, W., Schönemann, H., Siebert, T. (1995). Standard bases, syzygies, and their implementation in SINGULAR, Preprint 251, University of Kaiserslautern.

Grayson, D., Stillman, M. (1993–1998). Macaulay2: a system for computation in algebraic geometry and commutative algebra, `http://www.math.uiuc.edu/Macaulay2`, computer software.

La Scala, R. (1994). An algorithm for complexes, *Proceedings of ISSAC 94*, pp. 264–268. Oxford, ACM Press.

La Scala, R. (1996). Un approccio computazionale alle risoluzioni libere minimali, PhD Thesis, University of Bari.

Möller, H. M., Mora, T., Traverso, C. (1992). Gröbner bases computation using syzygies, *Proceedings of ISSAC 92*, pp. 320–328, Oxford, ACM Press.

Reisner, G. (1976). Cohen–Macaulay quotients of polynomial rings. *Adv. Math.*, **21**, 30–49.

Richman, F. (1974). Constructive aspects of noetherian rings, *Proc. Amer. Math. Soc.*, **44**, 436–441.

Schreyer, F.O. (1980). Die Berechnung von Syzygien mit dem verallgemeinerten Weierstrass'schen Divisionssatz, Diplomarbeit, Hamburg.
Siebert, T. (1996). On strategies and implementations for computations of free resolutions, Preprint 8, University of Kaiserslautern.
Spear, D. (1977). A constructive approach to commutative ring theory, *Proceedings of 1977 MACSYMA Users' Conference*, pp. 369–376. NASA CP-2012.
Yan, T. (1998). The geobucket data structure for polynomials, *J. Symb. Comput.*, **25**, 285–293.

Originally received 25 July 1997
Accepted 22 October 1997