On an implementation of standard bases and syzygies in SINGULAR

*Grassmann, H.; Greuel, G.-M.
Martin, B.; Neumann, W.;
Pfister, G.; Pohl, W.;
Schönemann, H.; Siebert, T.

July 17, 1995

Introduction

The use of Gröbner basis computations for treating systems of polynomial equations has become an important tool in many areas, reaching from pure mathematics to industrial applications. Despite the well–known complexity of Buchberger's algorithm, it has turned out to be very practicable for many special cases of interest. Actually, several of these applications became possible because of an improvement to certain strategies, for instance which pairs can be discarded or which should be considered first in the algorithm (cf. [8], [9]).

In contrast, very little is known about good strategies for computing standard bases in the local case, that is for monomial orderings which are not well orderings. The correctness and termination of a standard basis algorithm for such general semigroup orderings (a generalization of Buchberger's algorithm and of Mora's tangent cone algorithm) has been proved in [10]. In this article we describe its implementation in the system SINGULAR and present algorithms and strategies which have already proved quite useful in making possible standard basis computations in the local case, which were impossible by other means.

The strategies are based on heuristics and on a lot of experiments over the last years. Following tradition, we describe the implementation in a kind of pseudo code, explain the heuristics and justify our conclusions by a set of examples (which we have chosen in a representative manner from a much larger set). A publication of these methods would seem to be useful because of the practical importance of the standard basis algorithm and the lack of better methods.

Besides standard bases we compare different methods for computing syzygies. Syzygetic methods have grown to be an important tool in constructive module theory (cf. [6]) and the need for efficient algorithms seems to be important, at least in mathematical applications. We present experiments with different methods and with the first implementation of Schreyer's algorithm (cf. [19], [10]).

The implementation of the standard basis algorithm in SINGULAR and the strategies are described in §1. The most relevant new methods are the HCtest and the ecartMethod. The HCtest has dramatic impact on the efficiency of the algorithm for zero–dimensional ideals in the local case. It was first found by Pfister and Schemmel and has been implemented in a forerunner of SINGULAR since 1985. It consists of the computation of the minimal monomial not contained in the initial ideal and discarding all bigger monomials in further computations. The HCtest has also been used and proven to be fairly successful in an early implementation of the tangent cone algorithm by Zimnol [20] (which was based on ideas of H. Brakhage).

The ecartMethod consists of the choice of a weight vector of positive integers such that the weighted ecart of the input polynomials, with respect to this vector, become as small as possible. Of course, this method is most useful if the system offers an automatic choice of the ecart vector. In SINGULAR the automatic choice is realized by optimizing a certain functional but the user also has the option to choose his own ecart vectors. The ecartMethod and the functional is described in more detail in [18], together with another method to compute weights for the variables, which applies in the classical Buchberger case.

***Acknowledgement:** The authors were partially supported by the VW-Stiftung, the Stiftung für Innovation Rheinland–Pfalz, the DFG and the ESPRIT BRA contract 6846 POSSO.

Whilst the HCtest applies only to 0–dimensional ideals, the ecartMethod works in any case and is usually superior to other strategies. But we noticed that computing a standard basis with respect to the lexicographical ordering can be very fast in the local case. However, the meaning of this ordering is less clear (it is not an elimination ordering in this case). Nevertheless, it can be used to compute a lexicographical standard basis and then to recompute a standard basis from this one with respect to the desired ordering. Other methods include "morePairs", which has the effect of keeping some useless pairs in order to have better candidates for reduction, and include combinations of different methods. §2 contains a comparison of the different strategies by giving the timing for many examples. It should be emphasized that, for local and mixed orderings, it is essential to run a lazy version of the algorithm (cf. [17]) in order to have a good performance.

The algorithm of F.O. Schreyer to compute resolutions is implemented in SINGULAR. It is the first implementation even in the classical Buchberger case. We believe that this algorithm, and further improvements, will have theoretical as well as practical advantages in many cases of interest, in particular in cases which are far from a complete intersection. On the other hand, it does not give a minimal resolution and computing a minimal resolution step by step is sometimes more efficient. We include a table of experiments with comparisons of this algorithm to other methods in the classical homogeneous case.

SINGULAR is implemented in C and C++. A test version is available unter ftp from www.mathematik.uni-kl.de. The ground fields for standard basis computations are $\mathbb{Z}/p\mathbb{Z}$ (p a small prime number $< 2^{16}$), $\mathbb{Q}$, $\mathbb{Q}[Z]/(f)$, $(\mathbb{Z}/p\mathbb{Z})[Z]/(f)$ (f irreducible), $\mathbb{Q}(Z, Y, X, \ldots)$ and $(\mathbb{Z}/p\mathbb{Z})$ $(Z, Y, X, \ldots)$ (finitely many parameters). Of course, the computations in algebraic or transcendental extensions are very time consuming. We should like to mention, however, that there is only one standard basis algorithm implemented, which covers all possible semigroup orderings and all coefficient fields. Hence, the speed is due to the strategies and not to special tricks of the implementation.

1 The standard basis algorithm

In SINGULAR the standard basis algorithm is implemented as follows:

$S :=$ **StandardBasis (G)**
INPUT: G, a set of polynomial vectors $\subset K[x]^r$
OUTPUT: S, a standard base of the submodule generated by G with respect to the given monomial ordering

```
S := G^h, the homogeneization of G with respect to t.
update (S)
HCtest
T := S
L := initPairs (S)
clear (S)
WHILE  L ≠ ∅  DO
    p := (a, b, s) the last element from L
    L := L\{(a, b, s)}
    IF the spoly of a and b is not computed yet THEN
        s := spoly (a, b)
    END
    h := LazyNF (p|T)
    IF  h ≠ 0  THEN
        HCtest
        updatePairs (h)
        S := S ∪ {h}
        clear (S)
        T := T ∪ {h}
    END
END
S := completeReduce (S).
```

Remark 1.1 In cases when either $<$ is a wellordering, or $G \subseteq K[x]$ and $\dim_K K[x]/\langle G\rangle < \infty$ the standard basis is uniquely determined.

During the algorithm the set S is sorted by increasing monomial ordering of the leading terms of the elements with respect to the ordering in $K[x]$ (that is t is considered as parameter here).

The set T (respectively L) is sorted by increasing (respectively decreasing) monomial ordering of the leading terms of the elements (respectively the S-polynomial of the corresponding pair) with respect to the following ordering: $t^p x_1^{\alpha_1} \ldots x_n^{\alpha_n} > t^q x_1^{\beta_1} \ldots x_n^{\beta_n}$ if either $p + w_1\alpha_1 + \cdots + \alpha_n\alpha_n > q + w_1\beta_1 + \cdots + w_n\beta_n$ ($w_i > 0$ for all i suitable weights) or, in case of equality, $x_1^{\alpha_1}, \ldots, x_n^{\alpha_n} > x_1^{\beta_1}, \ldots, x_n^{\beta_n}$ with respect to the monomial ordering in $K[x]$.

Now we explain the procedures used in the standard basis algorithm:

Update (S)
INPUT: S, a set of polynomial vectors
OUTPUT: S, the set of polynomial vectors after the interreduction
for all $s \in S$ DO $s :=$ NFBuchberger $(s \mid S\backslash\{s\})$.

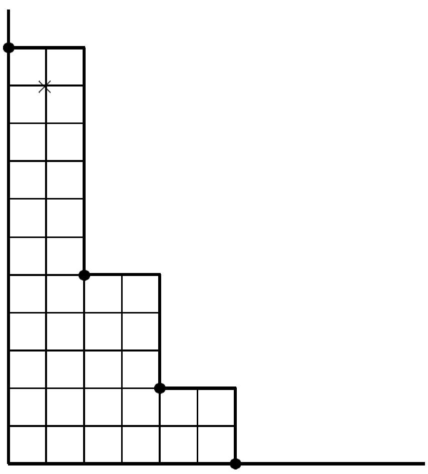
We call S **interreduced** if it is treated by Update (S).

HCtest
INPUT: S, a set of polynomials
OUTPUT: a boolean value (does a "highest corner" exist?) and, if so, the monomial "highest corner"
If $G \subseteq K[x]$ (that is $r = 1$) then it tests if there are a_i, $\alpha_i \geq 0$ such that $t^{a_i} x_i^{\alpha_i}$ occur as leading terms in S for all i. If this is true it computes the minimal monomial x^α (with respect to the ordering $<$ in $K[x]$) which is not in the ideal generated by the leading terms of the elements of $S(t=1)$. This monomial is called ''highest corner''.
It changes the polynomial arithmetic to cancel all monomials $t^a x^\beta$ with $x^\beta < x^\alpha$ in further computations.

Remark 1.2 The **highest corner** $\mathrm{HC}(I)$ of a 0–dimensional ideal $I \subset K[x]$ with respect to an ordering $<$ is the smallest monomial not contained in I. Obviously, the highest corner of I is equal to the highest corner of the leading ideal $L(I)$. If $I' \subset I$ are two ideals, then $\mathrm{HC}(I') \leq \mathrm{HC}(I)$. Since $\mathrm{HCtest}(S) = \mathrm{HC}(\langle L(S)\rangle)$, we have $\mathrm{HCtest}(S) \leq \mathrm{HC}(I)$ where $I = \langle S\rangle$ and $\mathrm{HCtest}(S) = \mathrm{HC}(I)$, if S is a standard basis of I. This proves the correctness of the algorithm.

Notice that the highest corner x^α is equal to 1 in the case of a wellordering. Therefore, the procedure is called only if $x_i < 1$ for some i. The highest corner for a set of monomials is computed in a combinatorial manner and its implementation is not described here.

The highest corner:



$\bullet$ correspond to leading terms of $S(t=1)$, $\times$ corresponds to the highest corner x^α.

L := **initPairs** (**S**)
INPUT: S, a set of (interreduced) polynomial vectors
OUTPUT: L, the set of critical pairs

It creates the pairset $L = \{(f, g, s) \mid f, g \in S,\ s$ the leading term of spoly$(f, g)\}$. Using the criteria similar to Gebauer-Möller (cf. [8]) useless pairs are cancelled.
We have two options: usually the criteria are applied for the pairs $\{(f(t = 1),\ g(t = 1)) \mid f, g \in S\}$, that is in $K[x]$. In the option sugar crit we apply it to L using the idea of [9].

Clear (**S**)
INPUT and OUTPUT: S, a set of polynomial vectors

Deletes f from S if $L(g) \mid t^{\alpha} L(f)$ for some α and $g \in S$.

Update (**h**)
INPUT and OUTPUT: h, a polynomial vector

```
IF  sugar  THEN RETURN END
( sugar is the strategy proposed in [9])
IF  t|h  THEN
    choose α maximal such that t^α|h
    h := h/t^α
END
```

UpdatePairs (**h**)
INPUT: h, a polynomial vector
S, a set of polynomial vectors
OUTPUT: L, the set of critical pairs from S and h

It updates the pairset L.
$L := L \cup \{(f, h, s) \mid f \in S,\ s$ the leading term of spoly$(f, h)\}$.
The criteria to cancel useless pairs are used as in initPairs.

h := **LazyNF** ((**a**, **b**, **s**) | **T**)
INPUT: s, a polynomial vector to reduce
(a, b), the critical pair from which s is the spoly
T, a set of polynomials with which to reduce
OUTPUT: h, the reduced polynomial vector

```
h := s
WHILE exist f ∈ T such that L(f) | t^α L(h) for some α  DO
      choose the first possible f with respect to the ordering in T
      such that α is minimal.
      IF  α > 0  THEN
           IF the position of (a,b,h) in L is not the last one THEN
              L := L ∪ {(a,b,h)}
              RETURN 0
           END
           T := T ∪ {h}
           NFupdate pairs (h)
      END
      h := spoly(t^α h, f)
      update (h)
      IF degree (h) > degree(s)   and
      the position of (a,b,h) in L is not the last one THEN
           L := L ∪ {(a,b,h)}
```

```
            RETURN 0
        END
    END
```

NFupdatePairs (h)
INPUT: h, a polynomial vector
S, a set of polynomial vectors
OUTPUT: L the set of some critical pairs from S and h

`IF NOT more pairs THEN RETURN END`
$L := L \cup \{(f, h, s) \mid f \in S,\ \deg_t L(f) \leq \deg_t L(h),\ s$ `the leading term of` spoly$(f, h)\}$
`The criteria to cancel useless pairs are used as in initPairs.`

Remark 1.3 The option **morePairs** (= `more pairs` in NFupdatePairs) has the effect of keeping some useless pairs in order to have better candidates for reduction. The leading idea is to keep (some of) those pairs which could not be discarded if we had used Lazard's method. In the case of a non–wellordering this has turned out to be useful, since it can help in not creating polynomials with too long tails during the normal form computation.

S := completeReduce (S)
INPUT and OUTPUT: S, a set of polynomial vectors

`IF` $<$ `is a wellordering or` $(S \subseteq K[x]$ `and` $\dim_K K[x]/\langle S\rangle < \infty)$
`THEN`

$S := S(t = 1)$
`FOR all` $s \in S$ `DO`
$s :=$ `redtail` $(s|S)$
`END`

`ELSE`

`FOR all` $s \in S$ `DO`
$s :=$ `redtail` $(s|S)$
`END`
$S := S(t = 1)$

`END`

s := redtail (s|S)
INPUT and OUTPUT: s, a polynomial vector
`reduces the monomial below` $L(s)$ `with elements from` S `as long as possible.`

2 Examples and comparisons

Let

$$\begin{array}{lcl} f_t^{a,b,c} & := & x^a + y^b + z^{3c} + x^{c+2}y^{c-1} + x^{c-1}y^{c-1}z^3 + x^{c-2}y^c(y^2 + tx)^2 \\ h^a & := & x^a + y^a + z^a + xyz(x + y + z)^2 + (x + y + z)^4 \\ g_t^{a,b,c,d,e} & := & x^a + y^b + z^c + x^d y^{e-5} + x^{d-2}y^{e-3} + x^{d-3}y^{e-4}z^2 + x^{d-4}y^{e-4}(y^2 + tx)^2 \end{array}$$

These three series stem from our attempts to disprove Zariski's conjecture (cf. [10]).

Examples for standard bases:

1. Alex 1
   $I = (5t^3x^2z + 2t^2y^3x^5,\ 7y + 4x^2y + y^2x + 2zt,\ 3tz + 3yz^2 + 2yz^4)$
2. $I = (xy + xz + 2xu + yu + zu + 2u^2 + yv + zv + 2uv - 3txz - y^2z - yz^2 - 3tzu - 2yzu - 3tzv + 3tyz^2,\ y^2 + z^2 + 2u^2 - 3u^2v,\ tx + z^2 + 2xv - txy,\ -yz - z^2 - xu - u^2 + tv - v^2,\ -1569xy^3 - 753t^2zu, -587xz + 15625yv)$
3. $f = f_1^{13,12,3},\ I = \left(\frac{\partial f}{\partial x},\ \frac{\partial f}{\partial y},\ \frac{\partial f}{\partial z}\right)$
4. $I = (2yz + 2xu + t^2x,\ 2tx+z^2+2yu+x^2y,\ x^2+2ty+yz+2zu,\ 2xy+2tz+u^2+z^3,\ y^2+2xz+2tu+u^5)$
5. $f = h^8,\ I = \left(\frac{\partial f}{\partial x},\ \frac{\partial f}{\partial y},\ \frac{\partial f}{\partial z}\right)$
6. $f = g_0^{6,8,10,5,5},\ I = \left(f, \frac{\partial f}{\partial x},\ \frac{\partial f}{\partial y},\ \frac{\partial f}{\partial z}\right)$
7. $J = (17y^6 + 49y^5 - 9y^4 + 12y^3 + 33y^2 + 11y + 73x^2z,\ 21x^2y + 17x^2z + 63y^3 + 11xz + 77y^2 + 91y + xz^2,\ 26y^2 + 44xz + 12y + 9zxy)$
   $I = J^2 + x^2z^2J$
8. $I = (bcd - u^5,\ y^2b^2c^2 - 2yzu^4bc + z^2u^8,\ t^2a^2b^2 - 2tyzu^3ab + y^2z^2u^6,\ txc - y^3zu,\ zwa - y^4u,\ xuw - y^5)$
9. $f = f_1^{13,12,3},\ I = \left(f, \frac{\partial f}{\partial x},\ \frac{\partial f}{\partial y},\ \frac{\partial f}{\partial z}\right)$
10. $I = (-x^{28}yz + t^{31} - t^{30}x,\ -xy^{25}z + t^{28} + t^{27}y,\ -xyz^{24} + t^{27} + t^{26}z)^2$
11. $I = (x^2 + y^2 + z^2 - 2yz^2 + y^4 + xy^2z,\ -2yz^2 + y^4 + xy^2z + x^3yz - y^2z^3,\ y^2z + x^3y - 3y^3z - 2xyz^2)^2$
12. Gräbe
    $I = (x^2 - z^{10} - z^{20},\ xy^3 - z^{10} - z^{30},\ y^6 - xy^3w^{40})$
13. $I = (txz + xyz + xy^2z + xyz^2,\ t^2z + xz^2 + yz^2 + x^2yz + xy^2z + xyz^2,\ t^2x + t^2z + xz^2 + x^2yz + xy^2z + xyz^2,\ txv + xuv + xu^2v + xuv^2,\ t^2v + xv^2 + uv^2 + x^2uv + xu^2v + xuv^2,\ t^2x + t^2v + xv^2 + x^2uv + xu^2v + xuv^2)$
14. $J = (-3x^2 - t^3 + x^3 - y^4,\ ty + 5x^3 - 3t^4,\ -20t^2 + x^2 - y^2)$
    $I = J^3 + zJ^2 + z^2J$
15. Random 1
    $I = (47x^7y^8z^3 + 91x^7y^4z^7 + 28x^3y^6z^8 + 63x^2y,\ 21x^3y^2z^{10} + 57xy^7z + 15x^3yz^5 + 51xy^3z^3,\ 32x^7y^4z^8 + 53x^6y^6z^2 + 17x^3y^7z^2 + 74xy^5z,\ 32x^{10}y^9z^6 + 23x^5y^8z^8 + 21x^2y^3z^7 + 27y^5z,\ 81x^{10}y^{10}z + 19x^3y^5z^5 + 79x^5z^7 + 36xy^2z^3)$
16. $I = (t^{18}x^2y - t^{19}z - t^{18}z^2,\ t^{26}xy - t^{27}z - t^{25}z^3,\ t^{38}y^2 - t^{37}xy)^2$
17. Random 2
    $I = (x^3 + y^4 + 2xz^3 + z^5 - 3x^4y^2 + 2z^6 + 3z^7,\ xz^3 - 2x^4y^2 + z^6 + 2z^7,\ 9x^3z^2 + 18x^2z^5 - 5z^7 + 12x^4y^2z^2 + 42x^2z^6 + 40x^3y^2z^4 + 7z^9 + 24x^3y^2z^5,\ -4y^3z^3 - 12x^6y + 32x^3y^5 - 2x^4yz^3,\ 12xy^3z^2 + 6x^5yz^2 + 24y^3z^5 + 20x^4yz^4 + 56y^3z^6 + 12x^4yz^5)$
18. Alex 2
    $I = (4t^2z + 6z^3t + 3z^3 + tz, 5t^2z^7y^3x + 5x^2z^4t^3y + 3t^7, 6zt^2y + 2x^8 + 6z^2y^2t + 2y^5)$
19. $f_1 = 2t^2 + x^2 + 2z^2 + 2u^2 + 2y^3,$
    $f_2 = 2tx + yz + 2tu + 2zu - t^3,$
    $f_3 = t^2 + 2tz + 2xu + 2yu - u^3,$
    $f_4 = 2ty + 2xz + 2tu - z^3,$
    $f_5 = x + 2y + 2z + 2u$
    $I = (f_1f_2,\ f_2f_3,\ f_3f_4,\ f_4f_5,\ f_5f_1)$
20. $f = f_1^{24,23,6}, I = \left(\frac{\partial f}{\partial x},\ \frac{\partial f}{\partial y},\ \frac{\partial f}{\partial z}\right)$

Options:
The ordering of the variables is $t, x, y, z, u, v, w, a, b, c, \ldots$, the computations were done in characteristic 32003. Lazard stands for Lazard's method (homogenization) and Mora for the standard basis algorithm described in §1 (see also [10]).
The columns have the following meaning:

1. Lazard with MACAULAY, elimination order for homogenizing variable (MAC)
2. Lazard with SINGULAR, elimination order for homogenizing variable (SING)
3. Mora with degrevlex$^-$ (–)
4. Mora with morePairs, degrevlex$^-$ (mP)

5. Mora with sugar and morePairs, degrevlex$^-$ (su/mP)
6. Mora with sugar, sugarCrit and morePairs, degrevlex$^-$ (suC/mP)
7. Mora with sugar, degrevlex$^-$ (su)
8. Mora with fast HCtest, degrevlex$^-$ (fHC)
9. Mora with weighted ecartMethod and sugar, but without HCtest, degrevlex$^-$ (ecartM).
   w denotes the ecart vector, if $w = (1, \ldots, 1)$ this is the same as 3.
10. Mora with lex$^-$ (lex)
    The orderings degrevlex$^-$ and lex$^-$ are described in [10].

We used the test version 0.9 of SINGULAR (May, 1995) and the version 3.0 of MACAULAY (July, 1994) for the tests on HP–UX, 9000/735 with 96 megabyte of main memory.

The symbol "#" indicates usage of more than 120 megabyte and "-" indicates that this option does not make sense. * means degree bound exceeded in Macaulay. The time is given in seconds. The time 0 indicates less than 0.5 seconds.

Compare the following table:

var denotes the number of variables, min the number of a minimal system of generators, std the number of elements in the standard base and dim the dimension of the zero–set of the ideal.

	MAC	SING	–	mP	su/mP	suC/mP	su	fHC	ecartM	lex	var	dim	min	std
1.	42	32	27	27	21	#	22	-	0	0	4	2	2	2
2.	509	719	303	474	462	466	251	-	251	117	6	1	6	40
3.	145	111	88	33	125	47	52	1	1	0	3	0	3	22
4.	84	66	17	29	20	30	12	-	4	18	5	1	5	21
5.	38	30	47	40	33	25	43	2	3	10	3	0	3	21
6.	10	7	23	4	3	4	20	0	0	0	3	0	4	17
7.	2	1	19	19	16	3	16	-	11	0	3	1	3	3
8.	*	50	20	92	89	65	19	-	19	33	10	6	6	1203
9.	15	9	281	6	5	4	218	0	0	0	3	0	4	21
10.	30	45	122	13	9	15	8	-	7	284	4	2	6	64
11.	5	4	22	2	2	2	17	-	*	2	3	1	6	9
12.	3	1	#	#	504	0	#	-	0	0	4	1	3	6
13.	25	24	12	12	9	7	9	-	8	27	6	3	6	22
14.	110	57	72	123	77	83	49	-	3	2	4	1	19	29
15.	1964	1390	0	0	0	0	0	-	0	0	3	1	3	4
16.	102	88	39	63	45	42	27	-	27	0	4	3	6	17
17.	3	3	13	1	1	1	11	-	0	0	3	0	5	8
18.	3531	3398	1	0	0	#	0	0	0	0	4	1	3	7
19.	464	646	16	755	543	643	11	-	3	19	5	2	5	16
20.	75961	46250	#	27931	28103	8032	#	641	121	115	3	0	3	44

The weights for ecartM are:

1.: (5 2 2 1),
2.: (1 1 1 1 1 1),
3.: (91 92 61),
4.: (2 2 2 2 1),
5.: (2 1 1),
6.: (16 8 5),
7.: (75 64 75),
8.: (1 1 1 1 2 2 3 4 4 2),
9.: (36 39 25),
10.: (15 16 16 16),
11.: (1 1 1),
12.: (20 20 2 1),
13.: (3 2 2 2 2 2),
14.: (32 32 24 117),
15.: (47 64 62),
16.: (1 1 1 1),
17.: (2 3 2),
18.: (32 12 19 13),
19.: (9 9 6 5 6),
20.: (76 65 47).

Conclusions: (for the case $Loc_< K[x] = K[x]_{(x)}$)
The above examples and many more tests which have not been documented show the following pattern:

1. The HCtest is essential for 0–dimensional ideals. In SINGULAR it is checked automatically whether all axes contain a leading monomial, regardless of whether the ideal is 0–dimensional or not (which is not very expansive). If this is so, the highest corner is computed and used as described in Chapter 1. In almost all cases this is the fastest option. Of course, it may be combined with the ecartMethod (and is so in SINGULAR if one chooses the ecartMethod). Moreover, if it is known in advance that the ideal is 0–dimensional we have the option fast HCtest (fHC) which does the following: in case all but one axes contain a leading monomial the search for a leading monomial on the last axes has priority. This (as for other options) is achieved by sorting the sets appropriately.

2. The ecartMethod applies to any ideal and can be extremely fast if one has a good feeling for the weights. In practice this option is most useful if the system automatically offers an ecart vector w. In SINGULAR this is implemented by choosing w such that the sum of the weighted ecart over all generators of the ideal (normalized in a certain way) is minimal. Other variants are under experimentation (cf. [18]).

3. The sugar option is generally good and is a default option in SINGULAR.

4. If the ecartMethod is not successful, it is usually recommended to use the option morePairs.

5. If the above options are not successful one should use morePairs and sugarCrit.

6. Lazard's method is in general slower than the other options, although it is sometimes surprisingly fast. In general it seems to be least a safe option (there are no #).

7. As the above timings show, the lex ordering is often the fastest. Since it is not an elimination ordering in the local case, its use is very limited (e.g. the dimension is computed correctly but not the multiplicity). But it can be used as a preprocessing in the sense that we first compute a standard bases with lex$^-$ and then transform this standard basis into one with the desired ordering, either by linear algebra methods or just by using it as an input to another standard basis computation. There are examples where this method, combined with the ecartMethod, is the only way to compute a standard basis for degrevlex$^-$.

These are general principles which have proved useful. Moreover, in SINGULAR the options can be combined by the user. Of course, there are always special examples with different behaviour. For example, in example 12 sugarCrit + morePairs is very good, but in example 1 it is the worst option. There is no option which is universally the best. The standard basis algorithm is extremely sensitive to the choice of the strategy, in the local case ($Loc_< K[x] = K[x]_{(x)}$) even more than in the inhomogeneous Buchberger case. The homogeneous Buchberger case, on the contrary, is much less sensitive to the choice of the strategy and is very stable. This explains why Lazard's method did always succeed, but on average is much slower. The ecartMethod (in connection with HC) seems to be by far the best. For most of the examples we used the automatic weight vector offered by SINGULAR, but for some we found a much better one by inspection of the input. In example 16 lex, together with d(lex), is the best (less than 0.1 seconds). The conclusion is that at the moment a system has to offer several strategies, a default one which is good in most cases, but also gives the user the possibility of another choice.

Moreover, for computations in characterstic zero or over algebraic extensions the growth of the coefficients has to be taken into account in the strategies.

We tried to compare the timings also with Faugère's GB, but GB neither offers the ordering used in Columns 1 and 2 nor the tangent cone orderings used in the remaining columns. Testing several homogeneous examples in positive characteristic with degrevlex ordering showed, however, that SINGULAR is about 1.4 times faster than GB (on a Sparc2 and on an IBM RS/6000).

Examples for resolutions:

1. the homogenization of Example 1 with respect to w
   (t, x, y, z, w the order of the variables)

2. the previous example with (w, t, x, y, z) being the order of the variables

3. the homogenization of Example 3 with respect to w
   (w, x, y, z the order of the variables)

4. the homogenization of Example 5 with respect to w
   (w, x, y, z the order of the variables)

5. the homogenization of Example 12 with respect to w
   (t, x, y, z, w the order of the variables)

6. the homogenization of Example 18 with respect to w
   (t, x, y, z, w)being the order of the variables

7. (max 5)^2
   the ideal $(f_1, \dots, f_5)^2$ where $f_i = x_i(x_0^8 + x_1^2 \cdot \dots \cdot \hat{x}_i \cdot \dots \cdot x_5^2)$, $i = 1, \dots, 5$.

8. Sparse
   $t^{18}x^2 - t^{19}z - t^{18}z^2$,
   $t^{26}xy - t^{17}z - t^{25}z^3$,
   $t^{38}y^2 - t^{37}xyw$;
   (t, x, y, z, w the order of the variables)

9. five homogeneous random polynomials of degree 3 in the variables a, b, c, d, e

10. cyclic roots 5
    $a + b + c + d + e$,
    $de + cd + bc + ae + ab$,
    $cde + bcd + ade + abe + abc$,
    $bcde + acde + abde + abce + abcd$,
    $abcde - h^5$;
    (a, b, c, d, e, h the order of the variables)

11. Kahn4
    $a^4 - b^4$,
    $b^4 - c^4$,
    $c^4 - d^4$,
    $d^4 - e^4$,
    $a^3b + b^3c + c^3d + d^3e + e^3a$;
    (a, b, c, d, e the order of the variables)

12. Iarrobino
    $xy + y^2 + uz - wz - xz + yz$,
    $x^2 - y^2 + uz - wz - xz + z^2$,
    wy,
    $wx - uz + yz$,
    $w^2 - y^2 + uz - z^2$,
    $vz - yz - z^2$,
    $vy - y^2 - wz + xz + yz + z^2$,
    $vx - y^2 + uz + yz - z^2$,
    $vw - y^2 + uz + yz - z^2$,
    $v^2 + y^2 + uz - xz - yz - z^2$,
    $uy + y^2 + yz$,
    $ux - y^2 + wz - xz - z^2$,
    $uw - y^2 + uz + yz - z^2$,
    $uv - y^2 - xz + yz$,
    $u^2 + yz$;
    (u, v, w, x, y, z the order of the variables)

13. three homogeneous random polynomials of degree 5 in the variables a, b, c, d, e

14. Schreyer 1
    polynomials created in a certain random manner such that the ideal generates a 0–dimensional Gorenstein ring in characteristic 31991.

$ab + 14766ac - 12713bc + 8997ad + 1878ae,$
$ac + 9210bc + 9399ad + 13903bd - 9583ae,$
$bc - 13988ad - 4712bd + 6771ae - 7341be,$
$ad - 2340bd - 7515cd + 1575ae - 4211be,$
$bd + 5023cd - 874ae + 4773be + 14016ce,$
$cd + 1617ae - 14718be - 1384ce + 12060de,$
$a^3 - 10731b^3 + 5818ae^2 + 13936be^2 + 11725ce^2 + 6401de^2,$
$b^3 - 4862c^3 - 2334ae^2 + 9991be^2 + 14579ce^2 + 10173de^2,$
$c^3 - 6087d^3 + 1135ae^2 + 4570be^2 + 5250ce^2 + 1393de^2,$
$d^3 + 11392ae^2 + 7125be^2 - 15188ce^2 - 12706de^2 - 10957e^3;$
(a, b, c, d, e the order of the variables)

15. Schreyer 2
polynomials created as in 14

$ab - 2302ac + 1504bc - 2175ad - 13468bd - 8487ae + 2984be,$
$ac + 7021bd - 2524ad - 8459bd + 4807cd - 4372ae - 1908be,$
$bc - 1768ad + 12994bd + 7816cd - 7083ae + 2769be - 354ce,$
$ad + 8219bd - 13765cd + 14989ae + 36be - 4732ce + 8648de,$
$a^3 - 7986a^2e - 744b^2e + 2389c^2e - 7991bde + 11075cde - 15425d^2e + 7645ae^2 - 1871be^2 + 8698ce^2 - 9609de^2,$
$a^2e + 3582b^2e - 6012c^2e - 14933bde + 14203cde + 11560d^2e + 2294ae^2 + 12826be^2 - 12284ce^2 + 10870de^2,$
$b^2e + 3097c^2e + 10611bde + 6826cde + 2785d^2e - 14231ae^2 + 13731be^2 - 1863ce^2 + 8227de^2,$
$c^2e - 8559bde + 13444cde + 3372d^2e + 5235ae^2 + 9139be^2 - 7685ce^2 + 3756de^2,$
$bde - 14077cde - 2960d^2e - 4481ae^2 - 3950be^2 - 850ce^2 - 14832de^2,$
$cde - 13603d^2e + 4327ae^2 + 15554be^2 + 2054ce^2 - 12484de^2,$
$b^3 + 7339d^2e - 3253ae^2 + 8720be^2 + 9224ce^2 - 11849de^2,$
$c^3 - 9222d^2e - 12818ae^2 - 11519be^2 + 6703ce^2 - 10364de^2,$
$d^3 + 4730d^2e + 7627ae^2 + 14168be^2 + 11428ce^2 + 8632de^2,$
$d^2e + 9376ae^2 + 6688be^2 - 11914ce^2 - 295de^2 - 5576e^3;$
(a, b, c, d, e the order of the variables)

16. Caprasse 4
$-2w^2x - w^2z + 2xyt + y^2z$
$2w^4 + 4w^2x^2 + 4w^2xz - 10w^2y^2 - 10w^2y^2 - 10w^2yt - x^3z + 4x^2yt + 4xy^2z + 2y^3t$
$-w^2x - 2w^2z + xt^2 + 2yzt$
$2w^4 + 4w^2xz - 10w^2yt + 4w^2z^2 - 10w^2t^2 - xz^3 + 4yz^2t + 2yt^3;$
(w, x, y, z, t the order of the variables)

17. Pellikaan, Jaworski S(9)
$S(m)$ denotes the set of polynomials (describing a minimal elliptic Gorenstein surface singularity), defined inductively:

$$\begin{array}{ll}
S(4): & x_4x_1 - x_2^2,\ x_4x_2 - x_3^2; \\
S(5): & S(4),\ f_{5,1}, f_{5,2}, x_5x_3 - x_4^2 + x_2x_1; \\
S(6): & S(5),\ f_{6,1}, f_{6,2}, f_{6,3}, x_6x_4 - x_5^2 - x_2^2; \\
S(m): & S(m-1),\ f_{m,1}, \ldots, f_{m,m-3}, g_m;\ \text{ for } m \geq 7, \text{ where} \\
f_{m,i} & = x_mx_i - x_{m-1}x_{i+2}, \\
g_m & = x_mx_{m-2} - x_{m-1}^2 - x_{m-3}x_{m-5},\ \text{ if } m \text{ is odd} \\
g_m & = x_mx_{m-2} - x_{m-1}^2 - x_{m-1}^2 - x_{m-2}x_{m-6};\ \text{ if } m \text{ is even}
\end{array}$$

($x_1, \ldots, x_m$ the order of the variables)

18. Pellikaan, Jaworski S(10)
compare example 17

19. 9 random i–forms in 4 variables,
$i = 2, \ldots, 10$

20. 6 random i–forms in 4 variables,
$i = 5, \ldots, 10$.

For these examples we computed a minimal resolution with MACAULAY and SINGULAR using the regularity bound (cf. [6]) with the ordering degrevlex (without this bound the computation is, at least for complicated examples, usually much slower) with two different strategies, and computed a resolution with Schreyer's method (SING/SCHREYER).

The first column, MAC, is the computation with Macaulay. The second column, Strat 1, computes a minimal resolution: after the computation of every module in the resolution the set of generators of this module is minimized.

The second strategy, Strat 2, uses selection strategies similar to Schreyer's method (cf. [10]) and minimizes during the computation of the resolution whenever possible.

SING/SCHREYER is the algorithm described in [10] as implemented in SINGULAR.

	MAC	SING Strat 1	SING Strat 2	SING SCHREYER	var	dim	min	std
1.	387	73	89	307	5	3	3	63
2.	1767	66	106	222	5	3	3	82
3.	12	4	2	2	4	1	3	75
4.	105	25	25	23	4	1	3	63
5.	8	1	2	2	4	2	3	83
6.	8	2	3	2	5	2	3	22
7.	8	3	3	163	5	4	15	138
8.	9	1	1	0	5	4	3	38
9.	10	5	9	9	5	1	5	76
10.	226	253	3	6	6	1	5	38
11.	719	108	116	116	5	0	5	142
12.	39	19	33	2	6	0	15	22
13.	36	12	18	23	5	2	3	25
14.	80	43	3	0	5	0	10	19
15.	60	19	9	1	5	0	14	23
16.	1	1	1	2	5	2	4	23
17.	1	2	3	1	9	2	27	27
18.	16	15	19	10	10	2	35	35
19.	9	6	3	2	4	0	6	31
20.	681	385	247	2080	4	0	6	192

Conclusions: The computation of more than 100 examples has shown that there is no best strategy. Minimizing whenever possible has the advantage of keeping the set of generators of a module small but it is not for free and has the disadvantage of sometimes creating more complicated generators.

Schreyer's algorithm computes a standard basis of each module in the resolution with respect to a special ordering depending on the generators of the preceding module (cf. [10]). Using this knowledge allows an implementation which makes the algorithm sometimes quite fast. On the other hand, since no minimizing is possible during the whole resolution, the set of generators may become huge, and, in this case, the algorithm becomes slow. This seems to happen if the ideal is close to a complete intersection (dim = var - min). Minimizing afterwards is possible in SINGULAR but usually not recommended for resolutions computed with Schreyer's method (it often takes more time than the whole resolution).

A comparison with the method proposed in [15] would be desirable.

It is possible that in the local case quite different methods turn out to be preferable. But this has not been investigated systematically so far.

References

[1] Bayer, D.; Stillman, M.: Macaulay. A computer algebra system for algebraic geometry.

[2] Becker, T.; Weispfenning, V.: Gröbner Bases. A computational approach to commutative algebra. Springer–Verlag GTM 141 (1991).

[3] Buchberger, B.: Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal. Thesis, Univ. Innsbruck, 1965.

[4] Buchberger, B.: Gröbner bases: an algorithmic method in polynomial ideal theory, in N.K Bose (ed.) Recent trends in multidimensional system theory, Reidel (1985).

[5] Eisenbud, D.: Commutative Algebra with a view toward Algebraic Geometry, 1995

[6] Eisenbud, D.: Commutative Algebra with a view toward Algebraic Geometry. Springer–Verlag, New York–Heidelberg–Berlin 1995.

[7] Gräbe, H.-G.: The tangent cone algorithm and homogenization. To appear.

[8] Gebauer, R.; Möller, M.: On an installation of Buchberger's Algorithm. J. Symbolic Computation (1988) **6**, 275–286.

[9] Giovini, A.; Mora, T.; Niesi, G.; Robbiano, L.; Traverso, C.: "One sugar cube, please" or selection strategies in the Buchberger algorithm. Proceedings of the 1991 ISSAC, 55-63.

[10] Greuel, G.-M.; Pfister, G.: Advances and improvements in the theory of standard bases and syzygies, to appear in Arch. of Math.

[11] Lazard, D.: Gröbner bases, Gaussian elimination, and resolution of systems of algebraic equations. Proc. EUROCAL 83, LN Comp. Sci. **162**, 146–156.

[12] Mora, T.: An algorithm to compute the equations of tangent cones. Proc. EUROCAM 82, Springer Lecture Notes in Computer Science (1982).

[13] Mora, T.: Seven variations on standard bases. Preprint, Univ. Genova (1988).

[14] Möller, H.M.; Mora, T.: Computational aspects of reduction strategies to construct resolutions of monomial ideals. Proc. AAECC 2, Lecture Notes in Computer Science **228** (1986).

[15] Möller, H.M.; Mora, T.: New Constructive Methods in Classical Ideal Theory. Journal of Algebra **100**, 138-178 (1986).

[16] Möller, H.M.; Mora, T.; Traverso, C.: Gröbner bases computation using syzygies. Proc. of ISSAC 1992.

[17] Mora, T.; Pfister, G.; Traverso, C.: An introduction to the tangent cone algorithm . Advances in Computing research, Issues in Robotics and nonlinear geometry (**6**) 199–270 (1992).

[18] Pohl, W.: About the weighted ecart and the weighted sugarMethod for computing standard bases. Preprint, Univ. Kaiserslautern, 1994.

[19] Schreyer, F.-O.: A standard basis approach to syzygies of canonical curves. J. reine angew. Math. **421**, 83-123 (1991).

[20] Zimnol, M.: Beispiele algebraischer Reduktionsstrukturen. Diplomarbeit, Kaiserslautern (1987).