Appendix A

Sheaf Cohomology and Beilinson Monads

In this appendix, which is more involved than the other parts of the book, we explain how to compute sheaf cohomology. As examples, we compute the cohomology of the ideal sheaves of the two surfaces constructed in Lecture 4. We also reveal some of the mathematics behind the constructions of the surfaces by discussing Beilinson monads. In our presentation, we follow Decker and Eisenbud (2002).

To fix our notation for what follows, let V be a vector space of dimension $n+1$ over a field K, $W = V^*$ its dual space, $\mathbb{P}^n = \mathbb{P}(W)$ the projective space of 1-quotients of W, and $S = \mathrm{Sym}_K(W)$ its homogeneous coordinate ring. We write $\mathcal{O}(i) = \mathcal{O}_{\mathbb{P}^n}(i)$ for the line bundles on $\mathbb{P}^n$ (in particular, $\mathcal{O} = \mathcal{O}(0)$ is the structure sheaf). If $\mathcal{F}$ is any coherent sheaf on $\mathbb{P}^n$, we write $\mathcal{F}(i) = \mathcal{F} \otimes \mathcal{O}(i)$ for the ith twist of $\mathcal{F}$, and $\mathrm{H}^j\mathcal{F} = \mathrm{H}^j(\mathbb{P}^n, \mathcal{F})$ for its jth cohomology group. Then

$$\mathrm{H}^j_*\mathcal{F} := \bigoplus_{i\in\mathbb{Z}} \mathrm{H}^j\mathcal{F}(i)$$

is a graded module over S. Note that Serre's sheafification functor $M \mapsto \widetilde{M}$ allows us to consider the coherent sheaf $\mathcal{F}$ as an equivalence class of finitely generated graded S-modules, where we identify two such modules M and M' if, for some r, the truncated modules $M_{\geq r}$ and $M'_{\geq r}$ are isomorphic.

Computing the cohomology of $\mathcal{F}$ could mean, for example, to compute one of the dimensions $\mathrm{h}^j\,\mathcal{F}(i) := \dim_K \mathrm{H}^j\mathcal{F}(i)$, or to compute these dimensions in a certain range of twists, or to compute the graded S-modules $\mathrm{H}^j_*\mathcal{F}$. We discuss two methods for this, one relying on the ability to compute free resolutions over the symmetric algebra S, and one asking for syzygy computations over the exterior algebra E on V.

The first method, described by Eisenbud (1998), is based on local duality:

Theorem A.1. *Let M be a finitely generated graded S-module, and let $\mathcal{F} = \widetilde{M}$ be the associated coherent sheaf. For all $j \geq 1$, we have*

$$\mathrm{H}^j_*\mathcal{F} \cong \mathrm{Ext}^{n-j}_S(M, S(-n-1))^\vee.$$

For $j = 0$, we have the exact sequence

$$0 \to \mathrm{Ext}_S^{n+1}(M, S(-n-1))^\vee \to M \to \mathrm{H}^0_* \mathcal{F} \to \mathrm{Ext}_S^n(M, S(-n-1))^\vee \to 0\,.$$

Here, if $N = \bigoplus_{i\in\mathbb{Z}} N_i$ is any graded S-module, we write $N^\vee$ for the graded vector space dual $N = \bigoplus_{i\in\mathbb{Z}} \mathrm{Hom}_K(N_{-i}, K)$ with its natural structure as a graded S-module.

Since we know how to compute Ext, the formula in the theorem allows us, in particular, to compute each of the dimensions $\mathrm{h}^j\,\mathcal{F}(i)$, starting from a given presentation of M. The `SINGULAR` command `sheafCoh` from `sheafcoh.lib` makes use of this idea.

If M is a finitely generated graded module over S, represented as $M = F/I$, where I is a graded submodule of a graded free module F over S, and if $l \le h$ are integers, then `sheafCoh(I,l,h)` makes `SINGULAR` display a cohomology table for the sheaf $\mathcal{F} = \widetilde{M}$ of the form

$$\begin{array}{cccc} \mathrm{h}^n\,\mathcal{F}(l) & \mathrm{h}^n\,\mathcal{F}(l+1) & \dots & \mathrm{h}^n\,\mathcal{F}(h) \\ \vdots & \vdots & & \vdots \\ \mathrm{h}^0\,\mathcal{F}(l) & \mathrm{h}^0\,\mathcal{F}(l+1) & \dots & \mathrm{h}^0\,\mathcal{F}(h) \end{array} \tag{A.1}$$

Example A.2. Continuing our `SINGULAR` session from Lecture 4, Example 4.13, we compute part of the cohomology of the ideal sheaf $\mathcal{F}$ of the Veronese surface in $\mathbb{P}^4$:

```
> LIB "sheafcoh.lib";
> module F = FI[2];
> int aa = timer;
> def B = sheafCoh(F,-2,6);
       -2   -1    0    1    2    3    4    5    6
-------------------------------------------------
  4:    -    -    -    -    -    -    -    -    -
  3:    3    -    -    -    -    -    -    -    -
  2:    -    -    -    -    -    -    -    -    -
  1:    -    -    -    1    -    -    -    -    -
  0:    -    -    -    -    -    7   25   60  119
-------------------------------------------------
chi:   -3    0    0   -1    0    7   25   60  119
> timer-aa;
2
```

The column with top entry i refers to the sheaf $\mathcal{F}(i)$. The bottom value in that column is the Euler-Poincaré characteristic of $\mathcal{F}(i)$,

$$\chi(\mathcal{F}(i)) = \sum_{i=1}^{n} \mathrm{h}^j\,\mathcal{F}(i)\,.$$

In our example, $n = 4$ and we have, for instance, $\chi(\mathcal{F}(1)) = -1$.

Similarly, we get for the surface constructed in Example 4.14:

```
> LIB "sheafcoh.lib";
> module F = FI[2];
> int aa = timer;
> def B = sheafCoh(F,0,8);
        0    1    2    3    4    5    6    7    8
-------------------------------------------------
  4:    -    -    -    -    -    -    -    -    -
  3:    -    -    -    -    -    -    -    -    -
  2:    -    2    -    -    -    -    -    -    -
  1:    -    -    -    2    -    -    -    -    -
  0:    -    -    -    -    1   15   47  105  198
-------------------------------------------------
chi:    0    2    0   -2    1   15   47  105  198
> timer-aa;
21
```

□

The second method we are going to discuss is due to Eisenbud, Fløystad, and Schreyer (2003). It makes use of a constructive version of the correspondence of Bernstein, Gelfand, and Gelfand (BGG for short). The BGG correspondence consists of a pair of adjoint functors $\mathbf{R}$ and $\mathbf{L}$ which define an equivalence between the derived category of bounded complexes of finitely generated graded S-modules and the derived category of bounded complexes of finitely generated graded E-modules (see Bernstein, Gelfand, and Gelfand (1978) for its original description).

Let us explain how $\mathbf{R}(M)$ is defined for a finitely generated graded S-module $M = \bigoplus_i M_i$, considered as a complex concentrated in cohomological degree 0. We grade S and E by taking elements of W to have degree 1 and elements of V to have degree -1. Then $\mathbf{R}(M)$ is the sequence of free E-modules and maps

$$\mathbf{R}(M): \quad \cdots \longleftarrow F^{i+1} \xleftarrow{\phi_i} F^i \xleftarrow{\phi_{i-1}} F^{i-1} \longleftarrow \cdots$$

defined as follows. Set

$$F^i = \mathrm{Hom}_K(E, M_i) = M_i \otimes_K \omega_E\,,$$

where

$$\omega_E = \mathrm{Hom}_K(E, K) = E \otimes \bigwedge^{n+1} W \cong E(-n-1)\,,$$

and where M_i is considered as a vector space concentrated in degree i. Further, let $\phi_i : F^i \to F^{i+1}$ be the map taking $\alpha \in \mathrm{Hom}_K(E, M_i)$ to

$$\Big(e \mapsto \textstyle\sum_j x_j \alpha(e_j \wedge e)\Big) \in \mathrm{Hom}_K(E, M_{i+1})\,,$$

where $\{x_j\}$ and $\{e_j\}$ are dual bases of W and V respectively. It is not too difficult to check that $\mathbf{R}(M)$ is indeed a complex.

An important fact is that this complex is eventually exact. The point at which exactness sets in is the Castelnuovo-Mumford regularity of M whose definition we recalled in Exercise 3.3. Alternatively, the regularity can be characterized as follows. If $M = \bigoplus_i M_i$ is a finitely generated graded S-module, then for all large integers r, the truncated module $M_{\geq r} \subset M$ is generated in degree r and has a **linear free resolution**; that is, its first syzygies are generated in degree $r+1$, its second syzygies in degree $r+2$, and so on. The Castelnuovo-Mumford **regularity** of M is the least integer r for which this occurs (see Eisenbud (1995), Chapter 20).

Theorem A.3 (Eisenbud, Fløystad, and Schreyer (2003)). *Let M be a finitely generated graded S-module, and let r be its Castelnuovo-Mumford regularity. The complex $\mathbf{R}(M)$ is exact at $\mathrm{Hom}_K(E, M_i)$ for all $i \geq s$ iff $s > r$.*

Starting from $\mathbf{T}^{>r}(M) := \mathbf{R}(M_{>r})$, we get a doubly infinite, exact, E-free complex $\mathbf{T}(M)$, the **Tate resolution of M**, by adjoining a minimal free resolution of the kernel of $\mathrm{Hom}_K(E, M_{r+1}) \to \mathrm{Hom}_K(E, M_{r+2})$. In fact, we may construct $\mathbf{T}(M)$ starting from any truncation $\mathbf{R}(M_{>s})$, $s \geq r$. So $\mathbf{T}(M)$ only depends on the sheaf $\mathcal{F} = \widetilde{M}$ associated to M. We write $\mathbf{T}(\mathcal{F}) = \mathbf{T}(M)$ and refer to this complex as the **Tate resolution of $\mathcal{F}$**.

Theorem A.4 (Eisenbud, Fløystad, and Schreyer (2003)). *Let M be a finitely generated graded S-module, and let $\mathcal{F} = \widetilde{M}$ be the associated coherent sheaf on $\mathbb{P}^n$. The term of the complex $\mathbf{T}(\mathcal{F})$ with cohomological degree i is*

$$\bigoplus_j H^j\mathcal{F}(i-j) \otimes \omega_E\,,$$

where $H^j\mathcal{F}(i-j)$ is regarded as a vector space concentrated in degree $i-j$.

Hence, each cohomology group of each twist of the sheaf $\mathcal{F}$ occurs exactly once in a term of $\mathbf{T}(\mathcal{F})$. We can, thus, compute part of the cohomology of $\mathcal{F}$ by computing part of the Tate resolution $\mathbf{T}(\mathcal{F})$.

The resulting algorithm is easy to implement – provided, the system we are working with offers the possibility of computing free resolutions over the exterior algebra. In `SINGULAR`, this is done by the kernel component `PLURAL` which we introduced in Lecture 3, Section 3.7.[1] The `SINGULAR` command `sheafCohBGG` from `sheafcoh.lib` makes use of `PLURAL` and the ideas discussed above. If M is a finitely generated graded module over S, represented as $M = F/I$, where I is a graded submodule of a graded free module F over S, and if $l \leq h$ are integers, then `sheafCohBGG(I,l,h);` prints a cohomology table for the sheaf $\mathcal{F} = \widetilde{M}$ of the form (A.1) on Page 274.

[1] Recall that `PLURAL` allows one to work over the large class of GR-algebras. At this writing, there is no custom-built, fast implementation specializing on the exterior algebra. As a result, the `SINGULAR` implementation of the algorithm of Eisenbud, Fløystad, and Schreyer is much slower than its `Macaulay2` implementation.

Example A.5. As in Example A.2, we continue our SINGULAR session from Lecture 4, Example 4.13, computing part of the cohomology of the ideal sheaf of the Veronese surface:

```
> LIB "sheafcoh.lib";
> module F = FI[2];
> int aa = timer;
> def B = sheafCohBGG(F,-2,6);
       -2   -1    0    1    2    3    4    5    6
 ------------------------------------------------
  4:    -    -    -    -    -    *    *    *    *
  3:    *    -    -    -    -    -    *    *    *
  2:    *    *    -    -    -    -    -    *    *
  1:    *    *    *    1    -    -    -    -    *
  0:    *    *    *    *    -    7   25   60  119
 ------------------------------------------------
chi:    *    *    *    *    0    *    *    *    *
> timer-aa;
70
```

Due to the shape of the Tate resolution, SINGULAR did not compute all dimensions $h^j\mathcal{F}(i)$ in the range $-2 \leq i \leq 6$. In the cohomology table printed above, the missing values are indicated by the symbol $*$. To compute also these values, enter `sheafCohBGG(F,-6,10);`. □

To explain some of the mathematics behind our construction of the surfaces considered above, we briefly discuss Beilinson monads. The technique of monads provides powerful tools for problems such as the construction and classification of coherent sheaves with prescribed invariants (see, for example, Okonek, Schneider, and Spindler (1980)). The basic idea is to represent arbitrary coherent sheaves in terms of simpler sheaves such as line bundles or bundles of differentials, and in terms of homomorphisms between these simpler sheaves.

Definition A.6. A **monad** on $\mathbb{P}^n$ with **homology** $\mathcal{F}$ is a bounded complex

$$\cdots \longleftarrow \mathcal{K}^1 \longleftarrow \mathcal{K}^0 \longleftarrow \mathcal{K}^{-1} \longleftarrow \cdots$$

of coherent sheaves on $\mathbb{P}^n$ with homology $\mathcal{F}$ at $\mathcal{K}^0$, and with no homology otherwise. □

If M is a finitely generated graded S-module, with associated sheaf $\mathcal{F} = \widetilde{M}$, the sheafification of the minimal free resolution of M is a monad for $\mathcal{F}$ which involves direct sums of line bundles and, thus, graded matrices over S. The Beilinson monad for $\mathcal{F}$ involves direct sums of twisted bundles of differentials and, thus, as we will see, graded matrices over E. It is much more directly connected with cohomology than the free resolution. Eisenbud, Fløystad, and Schreyer (2003) give an explicit construction of the Beilinson monad, starting from the Tate resolution.

To explain the construction, let $T^*_{\mathbb{P}(W)}$ be the cotangent bundle on $\mathbb{P}^n = \mathbb{P}(W)$, and let $\Omega^i = \Omega^i_{\mathbb{P}(W)} = \bigwedge^i T^*_{\mathbb{P}(W)}$ be the ith bundle of differentials (in particular, $\Omega^0 = \mathcal{O}$, $\Omega^n(n) = \mathcal{O}(-1)$, and $\Omega^i = 0$ if $i < 0$ or $i > n$). The fiber of $\Omega^1(1)$ at the point of $\mathbb{P}(W)$ corresponding to the line $\langle a \rangle \subset V$ is the subspace $(V/\langle a \rangle)^* \subset W$. Thus, $\Omega^1(1)$ fits as tautological subbundle into the short exact sequence

$$0 \longleftarrow \mathcal{O}(1) \longleftarrow W \otimes \mathcal{O} \longleftarrow \Omega^1(1) \longleftarrow 0 \ .$$

Taking exterior powers, we get the short exact sequences

$$0 \longleftarrow \Omega^i(i+1) \longleftarrow \bigwedge^{i+1} W \otimes \mathcal{O} \longleftarrow \Omega^{i+1}(i+1) \longleftarrow 0 \ .$$

Twisting the ith sequence by $-i-1$ and gluing them together, we get the exact sequence

$$0 \longleftarrow \bigwedge^0 W \otimes \mathcal{O} \longleftarrow \ \cdots \ \longleftarrow \bigwedge^{n+1} W \otimes \mathcal{O}(-n-1) \longleftarrow 0$$

which is nothing but the sheafification of the Koszul complex resolving $K = S/\langle W \rangle$. Thus, the bundles of differentials Ω^i are obtained as sheafifications of the syzygy modules $\mathrm{Syz}_{i+1}(K)$. By lifting homomorphisms between the bundles to homomorphisms between the Koszul resolutions, one shows (see, for example, Decker and Eisenbud (2002)):

Lemma A.7. *There are canonical isomorphisms*

$$\bigwedge^{i-j} V \xrightarrow{\cong} \mathrm{Hom}\big(\Omega^i(i), \Omega^j(j)\big), \quad 0 \le i, j \le n \, .$$

Under such an isomorphism, an element of $\bigwedge^{i-j} V$ *acts by contraction on the fibers of* $\Omega^i(i)$.

The construction of the Beilinson monad can now be easily described. Given $\mathbf{T}(\mathcal{F})$, we define $\Omega(\mathcal{F})$ to be the complex of vector bundles on $\mathbb{P}^n$ obtained by replacing each summand $\omega_E(i)$ by the sheaf $\Omega^i(i)$, and by using the isomorphisms

$$\mathrm{Hom}_E(\omega_E(i), \omega_E(j)) \cong \bigwedge^{i-j} V \cong \mathrm{Hom}(\Omega^i(i), \Omega^j(j))$$

to provide the maps.

Theorem A.8 (Eisenbud, Fløystad, and Schreyer (2003)). *Let* $\mathcal{F}$ *be a coherent sheaf on* $\mathbb{P}^n$. *Then* $\mathcal{F}$ *is the homology of* $\Omega(\mathcal{F})$ *in cohomological degree 0, and* $\Omega(\mathcal{F})$ *has no homology otherwise. We refer to* $\Omega(\mathcal{F})$ *as the* **Beilinson monad** *for* $\mathcal{F}$.

Since $\Omega^i = 0$ if $i < 0$ or $i > n$, only a small part of the Tate resolution and, thus, only a small part of the cohomology of $\mathcal{F}$ is actually involved in the construction of the Beilinson monad for $\mathcal{F}$. We are free, however, to apply the theorem also to twists of $\mathcal{F}$. In this way, we involve different parts of the cohomology and obtain, hence, different types of monads.

Example A.9. For the ideal sheaves of the surfaces in $\mathbb{P}^4$ considered in Example A.2, we get, for instance, the Beilinson monads

$$0 \longleftarrow \mathcal{J}_X(2) \longleftarrow \Omega^1(1) \longleftarrow \mathcal{O}(-1)^3 \longleftarrow 0 \ ,$$

respectively

$$0 \longleftarrow \mathcal{J}_Y(4) \longleftarrow \big(\Omega^1(1)\big)^2 \oplus \mathcal{O} \longleftarrow \big(\Omega^3(3)\big)^2 \longleftarrow 0 \ .$$

Indeed, this follows by inspecting the cohomology tables printed by the `sheafCoh` command in Example A.2. The astute reader will observe that in Lecture 4, Examples 4.13 and 4.14, we actually constructed the surfaces by constructing their Beilinson monads. See Decker, Ein, and Schreyer (1993) and Decker and Schreyer (2000) for the construction of smooth surfaces in $\mathbb{P}^4$. □

Remark A.10 (Further Reading). For more details and proofs of the results presented in this lecture, see Eisenbud (1998), Eisenbud, Fløystad, and Schreyer (2003), Decker and Eisenbud (2002), and Smith (2000).

Appendix B

Solutions to Exercises

In this appendix, we present solutions to all exercises posed in the practical sessions. We print the relevant SINGULAR code, but not the corresponding output.

Exercise 1.1. (a) On a Unix-like platform, start a `Singular` session by either entering `Singular` or `ESingular` in a command shell. In the first case, the session will be run in the command shell. In the second case, it will be run in an emacs buffer. Having started the session, enter

```
> ring R;
> R;
```

Now, you may define the polynomial f by typing

```
> poly f = x^4+x^3*z+x^2*y^2+y*z^4+z^5;
```

Alternatively, use the short format

```
> poly f = x4+x3z+x2y2+yz4+z5;
```

To display f, type `f;`.

(b) Proceed as follows:

```
> ring S = 32003, (x,y,z), lp;
> poly g = fetch(R,f);
> g;
```

To exit SINGULAR, type `quit;`, `exit;`, or `$`. Within emacs, preferably enter `CTRl-C $`. □

Exercise 1.2. As a finite field, we choose $\mathbb{F}_{32003}$ (in characteristic 0, the computation below will not terminate in due time).

```
> ring R = 32003, x(1..5), lp;
> int d = 5;
```

To define an ideal I generated by 10 homogeneous random polynomials of degree d, you may proceed as follows:

```
> ideal MId = maxideal(d);
> int s = size(MId);
> int i,j;
> ideal I;
> for (i=1; i<=10; i++)
. {
.   poly f(i);
.   for (j=1; j<=s; j++)
.   {
.     f(i) = f(i)+random(0,32002)*MId[j];
.   }
.   I = I, f(i);
. }
```

Alternatively, load the library `random.lib` and use one of its commands to create I:

```
> LIB "random.lib";
> ideal I = randomid(maxideal(d),10,32002);
```

(a)-(d) Proceed as follows:

```
> int aa = timer;
> ideal II = groebner(I);
> timer-aa;
> size(II);
> II;
> deg(II[1]);
> deg(II[size(II)]);
> write (":w lexGB.out",II);
```

(e) Proceed as follows:

```
> ring R1 = 32003, x(1..5), dp;
> ideal I = imap(R,I);
> aa = timer;
> ideal II = std(I);
> timer-aa;
> size(II);
> II;
> deg(II[1]);
> deg(II[size(II)]);
> write (":w dpGB.out",II);
```

□

Exercise 1.3. Define the matrix M and the ideal I:

```
> ring R = 0, x(0..4), dp;
> matrix M[2][4] = x(0),x(1),x(2),x(3),
.                  x(1),x(2),x(3),x(4);
> ideal I = minor(M,2);
```

(a) Compute the minimal free resolution and display the Betti diagram:

```
> resolution rI = mres(I,0);
> print(betti(rI),"betti");
```

(b) Display the syzygy matrices and determine their data type:

```
> for (int i=1; i<=3; i++)
. {
.   i,"th syzygy matrix: ";
.   "-------------------- ";
.   print(rI[i]);
.   "";
.   "has data type : ", typeof(rI[i]);
.   "";
. }
```

(c) Display the two representations of the Hilbert series:

```
> ideal GI = groebner(I);
> hilb(GI);
```

For the interpretation of the output, type `help hilb;` and follow the link to `Hilbert function`.

(d) To check smoothness, apply the Jacobian Criterion 2.23 (taking Corollary 2.26 into account):

```
> matrix JM = jacob(I);
> int codimI = nvars(R) - dim(GI);
> ideal singI = minor(JM,codimI) + I;
> nvars(R) - dim(groebner(singI));
```

□

Exercise 1.4. (a) Here is the desired procedure:

```
proc maximaldegree (ideal I)
"USAGE:  maximaldegree(I);   I=ideal
RETURN: integer; the maximum degree of the given
                 generators for I
NOTE:   return value is -1 if I=0
"
{
  if (size(I)>0)
  {
    int i,dd;
    int d = deg(I[1]);
    for (i=2; i<=size(I); i++)
    {
      dd = deg(I[i]);
      if (dd>d) { d = dd; }
    }
```

```
    return(d);
  }
  else
  {
    return(-1);
  }
}
```

(b) Having entered the procedure, apply it as follows:

```
> ring R = 32003, x(1..5), lp;
> string xxx = "ideal II="+read("lexGB.out")+";";
> execute(xxx);
> maximaldegree(II);
> xxx = "ideal JJ="+read("dpGB.out")+";";
> execute(xxx);
> maximaldegree(JJ);
```

□

Exercise 1.5. All smoothness tests are performed as in Exercise 1.3 (d) by applying the Jacobian criterion (all ideals are of pure codimension 2 by construction).

(a) To compute equations for the Veronese surface in $\mathbb{P}^5$, use `preimage`:

```
> ring P2 = 0, (u,v,w), dp;
> ideal emb = u2, v2, w2, uv, uw, vw;
> ideal I0 = ideal(0);
> ring P5 = 0, x(0..5), dp;
> ideal VP5 = preimage(P2, emb, I0);
> print(betti(list(VP5)),"betti");
```

(b) Define the ideal of the point $p = (0 : 0 : 0 : 1 : 1 : 1)$ and check that p does not lie on the Veronese surface in $\mathbb{P}^5$. Project the surface from p to $\mathbb{P}^4$:

```
> ideal p = x(0), x(1), x(2), x(3)-x(5), x(4)-x(5);
> size(reduce(VP5,groebner(p),1));
> ring P4 = 0, x(0..4), dp;
> ideal VP4 = preimage(P5,p,VP5);
> print(betti(list(VP4)),"betti");
```

(c) Randomly choose two $\mathbb{Q}$-linear combinations of the 7 cubics generating `VP4`:

```
> LIB "elim.lib"; // loads random.lib, too
> ideal CI1 = randomid(VP4,2,100);
```

To compute the ideal of the linked surface, saturate `CI1` with respect to `VP4`:

```
> ideal QES = sat(CI1,VP4)[1];
```

Compute the minimal free resolution and display the Betti diagram:

```
> resolution FQES = mres(QES,0);
> print(betti(FQES),"betti");
```

(d) Proceed as in (b) and (c) above, projecting from the point $q = (1 : 1 : 1 : 1 : 1 : 1)$:

```
> setring P5;
> ideal q = x(0)-x(1), x(1)-x(2), x(2)-x(3), x(3)-x(4), x(4)-x(5);
> size(reduce(VP5,groebner(q),1));
> setring P4;
> ideal CS = preimage(P5,q,VP5);
> print(betti(list(CS)),"betti");
> ideal CI2 = matrix(CS)*randommat(3,2,maxideal(1),100);
> ideal B = sat(CI2,CS)[1];
> print(betti(list(B)),"betti");
```

(e) Now, project from the line $\mathrm{V}(x_0 + x_1 + x_2, x_3, x_4, x_5)$:

```
> setring P5;
> ideal L = x(0)+x(1)+x(2), x(3), x(4), x(5);
> nvars(P5) - dim(groebner(VP5+L));
> ring P3 = 0, y(0..3), dp;
> ideal SRS = preimage(P5,L,VP5); SRS;
```

□

Exercise 2.1. (a) Use the command `algDependent` from `algebra.lib`:

```
> ring R = 0, (x,y), dp;
> poly f1, f2, f3 = x2+y2, x2y2, x3y-xy3;
> LIB "algebra.lib";
> def L = algDependent(ideal(f1,f2,f3));
```

Then follow the explanation printed by `SINGULAR`:

```
> L[1];
> def S = L[2];
> setring S;
> ker;
```

(b) Use the command `algebra_containment` from `algebra.lib`:

```
> setring R;
> poly g, g1, g2 = x4+y4, x+y, xy;
> L = algebra_containment(g,ideal(g1,g2),1);
```

Then follow the explanation printed by `SINGULAR`:

```
> def S2 = L[2];
> setring S2;
> check;
```

(c) Use the command `is_bijective` from `algebra.lib`:

```
> ring T = 0, x(1..3), dp;
> qring Q = groebner(x(1)*x(2)*x(3)-1);
> map phi = Q, x(2)*x(3), x(1)*x(3), x(1)*x(2);
> is_bijective(phi,Q);
```
□

Exercise 2.2. (a) Define the ideal of the affine twisted cubic curve and homogenize it with respect to a slack variable:

```
> ring R = 0, (w,x,y,z), dp;
> ideal I = y-x2, z-x3;
> ideal SI = groebner(I);
> ideal Ih = homog(SI,w);          // generators are homogenized
> Ih;
```

Since the generators for `I` do not depend on w, the computation of `SI` takes place in $\mathbb{Q}[x,y,z]$. The result is a degree reverse lexicographic Gröbner basis for `I` in $\mathbb{Q}[x,y,z]$. Homogenizing its elements yields generators for the homogenized ideal I^{hom} (in general, however, the result of such a computation is not necessarily a Gröbner basis for I^{hom} with respect to $>_{\mathtt{dp}}$ on $\mathbb{Q}[w,x,y,z]$, see Proposition 2.35).

(b) Define the ideal `J` obtained by homogenizing the original generators for `I`, check that `J` is strictly contained in `Ih`, and compute the "extra component at infinity" (which is a triple structure on the line $w = x = 0$):

```
> ideal J = homog(I,w);            // generators are homogenized
> size(reduce(J,groebner(Ih),1));  // J is contained in Ih
> size(reduce(Ih,groebner(J),1));  // Ih is not contained in J
> LIB "elim.lib";
> ideal Iinf = sat(J,Ih)[1];
> Iinf;
```
□

Exercise 2.3. Consider the "universal polynomial"

$$f = a_1x^3 + a_2x^2y + a_3xy^2 + a_4y^3 + a_5x^2 + a_6xy \\ + a_7y^2 + a_8x + a_9y + a_{10} \in \mathbb{C}\left[x, y, a_1, \ldots, a_{10}\right],$$

and let J be the ideal $J = \langle f, \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\rangle$. Then the Zariski closure of the desired locus in the affine space $\mathbb{A}^{10}$ with coordinates $a_1, \ldots, a_{10}$ is obtained as the image of $\mathrm{V}(J) \subset \mathbb{A}^2 \times \mathbb{A}^{10}$ under the projection onto $\mathbb{A}^{10}$. It is, thus, defined by the elimination ideal $J \cap \mathbb{C}[a_1, \ldots, a_{10}]$ which we compute directly following Proposition 2.30 (we do not use the commands `eliminate` and `preimage`). For this, we implement the ring $\mathbb{Q}[x, y, a_1, \ldots, a_{10}]$ equipped with a monomial order having the elimination property with respect to x, y.

```
> ring S = 0, (y,x), dp;
> ideal I = maxideal(3),maxideal(2),x,y,1;
> ring R = 0, (x,y,a(1..10)), (dp(2),dp(10));
```

```
> ideal I = imap(S,I);
> matrix A[10][1] = a(1..10);
> poly f = (matrix(I)*A)[1,1];
> ideal J = f, diff(f,x), diff(f,y);
> J = groebner(J);
> // check for generators for J that do not depend on x,y
. ideal JJ;
> int i;
> for (i=1; i<=size(J); i++) { if (J[i][1]<y){ JJ=JJ,J[i]; } }
> JJ = simplify(JJ,2);              // erase zero generators
> size(JJ);
> homog(JJ);
```

We see that the elimination ideal is generated by a single homogeneous polynomial. This polynomial has degree 12 and 2040 terms:

```
> poly D = JJ[1];
> deg(D); size(D);
```

Note that the defining polynomial $F_{\boldsymbol{a}}$ of a cubic curve in $\mathbb{P}^2$ is determined up to a scalar multiple. Thus, these curves are parametrized by the projective space $\mathbb{P}^9 = \mathbb{P}^9(\mathbb{C})$ associated to the vector space formed by the $F_{\boldsymbol{a}}$. Since the polynomial `D` is homogeneous, it defines a hypersurface $\mathrm{V}(D)$ in $\mathbb{P}^9$. What we just verified is that $\mathrm{V}(D)$ is the Zariski closure of the locus of points for which the corresponding cubic curve has a singular point in the affine chart $U_z = \{z \neq 0\}$ of $\mathbb{P}^2$. In Exercise 4.6, we will see that $\mathrm{V}(D)$ parametrizes precisely the cubic curves in $\mathbb{P}^2$ having a singular point p in $\mathbb{P}^2$ (in the sense that $F_{\boldsymbol{a}}$ and all its first partial derivatives vanish at p). We refer to `D` as the **discriminant** of the homogeneous "universal polynomial"

$$F = a_1x^3 + a_2x^2y + a_3xy^2 + a_4y^3 + a_5x^2z + a_6xyz \\ +a_7y^2z + a_8xz^2 + a_9yz^2 + a_{10}z^3\,. \qquad \square$$

Exercise 2.4. We collect the procedures in a text file, say `sol.lib`, respecting the recommended format for a `SINGULAR` library (see Page 112).

```
version = "1.0";
info = "solution to Exercise 2.4";
//
proc ideal_intersect (ideal I, ideal J)
"USAGE:   ideal_intersect(I,J);    I,J ideals
RETURN:  ideal
NOTE:    Output is generating set for the intersection of I and J.
EXAMPLE: example ideal_intersect; shows an example
"
{
  int r = size(I);
  int s = size(J);
```

```
  if ((r==0) or (s==0)) { return(ideal(0)); }
  module M = gen(1)+gen(2);
  for ( int i=1;i<=r;i++ ) { M = M, I[i]*gen(1); }
  for ( i=1;i<=s;i++ ) { M = M, J[i]*gen(2); }
  module S = syz(M);
  ideal result;
  for ( i=ncols(S);i>0;i-- ) { result[i] = S[i][1]; }
  return(simplify(result,2));      // remove zeros in result
}
example
{ "EXAMPLE:"; echo = 2;
  ring R = 0, (x,y), dp;
  ideal I = x2, y;
  ideal J = x, y2;
  ideal_intersect(I,J);
}
//
proc ideal_quotient (ideal I, ideal J)
"USAGE:   ideal_quotient(I,J);    I,J ideals
RETURN:  ideal
NOTE:    Output is generating set for the quotient I:J.
EXAMPLE: example ideal_quotient; shows an example
"
{
  int r = size(I);
  int s = size(J);
  if ((r==0)) { return(ideal(0)); }
  if ((s==0)) { return(I); }
  vector v;
  for ( int i=1;i<=s;i++ ) { v = v+J[i]*gen(i); }
  module M = v;
  for ( int j=1;j<=s;j++ )
  {
    for ( i=1;i<=r;i++ ) { M = M, I[i]*gen(j); }
  }
  module S = syz(M);
  ideal result;
  for ( i=ncols(S);i>0;i-- ) { result[i] = S[i][1]; }
  return(simplify(result,2));
  return(result);
}
example
{ "EXAMPLE:"; echo = 2;
  ring R = 0, (x,y), dp;
  ideal I = x2, y;
  ideal J = x, y2;
  ideal_quotient(I,J);
}
//
```

```
proc saturate (ideal I, ideal J)
"USAGE:   saturate(I,J);    I,J ideals
RETURN:  ideal
NOTE:    Output is generating set for the saturation of I with
         respect to J.
EXAMPLE: example saturate; shows an example
"
{
  ideal I_old = groebner(I);
  ideal I_new;
  while (1)
  {
    I_new = groebner(ideal_quotient(I_old,J));
    if (size(reduce(I_new,I_old))==0) { return(I_new); }
    I_old = I_new;
  }
}
example
{ "EXAMPLE:"; echo = 2;
  ring R = 0, (x,y), dp;
  ideal I = x5*(x-1), y3;
  ideal J = x, y2;
  saturate(I,J);
}
```

Having loaded this file into a SINGULAR session, the procedures are accessible. In particular, we may apply the example command to make SINGULAR run the examples (without affecting the current SINGULAR session):

```
> LIB "sol.lib";
// ** loaded sol.lib 1.0
> example ideal_intersect;
// proc ideal_intersect from lib sol.lib
EXAMPLE:
  ring R = 0, (x,y), dp;
  ideal I = x2, y;
  ideal J = x, y2;
  ideal_intersect(I,J);
_[1]=y2
_[2]=xy
_[3]=x2
```

□

Exercise 3.1. Here is the desired procedure:

```
proc is_reg_sequence (ideal I)
"USAGE:   is_reg_sequence(I);  I ideal,
RETURN:  1 if the given (ordered) list of generators for I is a
           regular sequence;
         0 otherwise.
"
```

```
{
  int i;
  ideal J;
  while(i<size(I))
  {
    i++;
    if (size(reduce(quotient(J,I[i]),J))!=0)
    {
      return(0);
    }
    J = groebner(J+I[i]);
  }
  if (size(reduce(1,J))==0) { return(0); }
  return(1);
}
```

We apply the procedure to the given (ordered) list of polynomials f_1, f_2, f_3 and the permutation f_1, f_3, f_2 thereof:

```
> ring R = 0, (x,y,z), dp;
> ideal I = (x-1)*z, (x-1)*y, x;
> is_reg_sequence (I);
0
> I = (x-1)*z, x, (x-1)*y;
> is_reg_sequence (I);
1
```

□

Exercise 3.2. As being Cohen-Macaulay is a local property (see Proposition 5.37 and Remark 5.38), and since each of the given affine rings R is graded, it suffices to check in each case whether the localization $R_{\mathfrak{m}}$ of R at the homogeneous maximal ideal $\mathfrak{m}$ is Cohen-Macaulay. Apply the command `isCM` from `homolog.lib`:

(a)
```
> LIB "homolog.lib";
> ring R1 = 0, (x,y,z), dp;
> ideal I =  xy, yz, xz;
> ring R1_loc = 0, (x,y,z), ds;
> ideal I = imap(R1,I);  // ideal generated by I in localized ring
> isCM(I);
```

(b)
```
> ring R2 = 0, (s,t,x,y,z,w), dp;
> ideal I =  x-s4, y-s3t, z-st3, w-t4;
> ideal IC = eliminate(I,st);
> ring R2_loc = 0, (x,y,z,w), ds;
> ideal IC = imap(R2,IC);
> isCM(IC);
```

□

Exercise 3.3. (a) We compute the truncated module $M_{\geq d}$ by applying `modulo`: let $R = K[x_1, \ldots, x_n]$, and let

$$0 \longleftarrow M \longleftarrow F_0 = \bigoplus_{i=1}^{s} R(-v_i) \xleftarrow{\varphi} F_1$$

be a (graded) free presentation of M. Moreover, let L be a matrix whose columns span $\bigoplus_{i=1}^{s} \langle \boldsymbol{x} \rangle^{d-v_i} \cdot e_i \subset F_0$, where $e_1, \ldots, e_s$ denote the canonical basis vectors of F_0. Then

$$M_{\geq d} = \big(\operatorname{im} L + \operatorname{im} \varphi\big) \big/ \operatorname{im} \varphi \,.$$

Here is the resulting procedure:

```
proc truncate(module phi, int d)
"USAGE:   truncate(phi,d);  phi module, d int
ASSUME:  phi comes assigned with an admissible degree vector as an
         attribute
RETURN:  module
NOTE:    Output is a presentation matrix for the truncation of
         coker(phi) at d.
"
{
  if ( typeof(attrib(phi,"isHomog"))=="string" )
  {
    ERROR("No admissible degree vector assigned");
  }
  else
  {
    intvec v=attrib(phi,"isHomog");
  }
  int s = nrows(phi);
  int i,m,dummy;
  module L;
  for (i=1; i<=s; i++)
  {
    if (d>v[i])
    {
      L = L+maxideal(d-v[i])*gen(i);
    }
    else
    {
      L = L+gen(i);
    }
  }
  L = modulo(L,phi);
  L = prune(L);
  if (size(L)==0) {return(L);}

  // it only remains to set the degrees for L:
  // ------------------------------------------
  m = v[1];
```

```
  for(i=2; i<=size(v); i++)
  {
    if(v[i]<m)
    {
      m = v[i];
    }
  }
  dummy = homog(L);
  intvec vv = attrib(L,"isHomog");
  if (d>m)
  {
    vv = vv+d;
  }
  else
  {
    vv = vv+m;
  }
  attrib(L,"isHomog",vv);
  return(L);
}
```

(b) For the Castelnuovo-Mumford regularity, we compute the number of rows in the Betti diagram corresponding to the minimal free resolution of M:

```
proc CM_regularity (module phi)
"USAGE:   CM_regularity(phi);    phi module
ASSUME:  phi comes assigned with an admissible degree vector as an
         attribute
RETURN:  integer
NOTE:    Output is the Castelnuovo-Mumford regularity of coker(phi).
"
{
  if ( typeof(attrib(phi,"isHomog"))=="string" )
  {
    ERROR("No admissible degree vector assigned");
  }
  def L = mres(phi,0);
  intmat BeL = betti(L);
  int r = nrows(module(matrix(BeL)));
  int shift = attrib(BeL,"rowShift");  // See Section 3.4
  return(r+shift-1);
}
```

(c) To begin with, define the module I (and, thus, the module $M = F/I$):

```
> ring R = 0, (w,x,y,z), dp;
> module I = [xz,0,-w,-1,0], [-yz2,y2, 0,-w,0], [y2z,0,-z2,0,-x],
.            [y3,0,-yz,-x,0], [-z3,yz,0,0,-w], [-yz2,y2,0,-w,0],
.            [0,0,-wy2+xz2,-y2,x2];
> homog(I);
```

After having read the procedures designed in part (a) and (b) into SINGULAR, apply them as desired:

```
> CM_regularity(I);
3
> def T2 = mres(truncate(I,2),0);
> print (betti(T2),"betti");
           0     1     2     3
------------------------------
    2:    19    36    23     6
    3:     -     -     1     -
------------------------------
total:    19    36    24     6

> def T3 = mres(truncate(I,3),0);
> print (betti(T3),"betti");
           0     1     2     3
------------------------------
    3:    40    91    71    19
------------------------------
total:    40    91    71    19
```

For a better understanding of what has been computed, compare the output with the characterization of the regularity given prior to Theorem A.3 in Appendix A.

Exercise 3.4. We present two procedures, one for computing kernels of module homomorphisms, and one for computing Ext modules. Since $\mathrm{Hom}_R(M,N)$ is isomorphic to $\mathrm{Ext}^0_R(M,N)$, the latter procedure can in particular be used to compute Hom. Again, we collect the procedures in a text file, respecting the recommended format for a SINGULAR library.

For the kernel, we compute the modules A and B described in the solution to the Image and Kernel Problem 4.3 on Page 133:

```
version = "1.0";
info = "solution to Exercise 3.4";
//
LIB "matrix.lib";
//
proc ker_Mod (matrix alp, module phi,psi)
"USAGE:   ker_Mod(alp,phi,psi);   alp matrix, phi,psi modules
RETURN:  module
NOTE:    The generators for the output module are the columns of a
         presentation matrix for the kernel of the homomorphism
         coker(phi)->coker(psi) induced by alp.
EXAMPLE: example ker_Mod; shows an example
"
{
  module A = modulo(alp,psi);
```

```
  module B = modulo(A,phi);
  return(B);
}
example
{ "EXAMPLE:"; echo = 2;
  ring R = 0, (x,y,z), dp;
  module phi = [y3,-xy2];
  module psi = [0,y,0],[0,0,z];
  module alp = [x+xy,z,0],[y+y2,xyz,z];
  print(ker_Mod(alp,phi,psi));
}
```

Now, we turn to Ext. To compute $\mathrm{Ext}^i_R(M,N)$ from given free presentations

$$0 \longleftarrow M \longleftarrow F_0 \xleftarrow{\varphi} F_1 \quad \text{and} \quad 0 \longleftarrow N \longleftarrow G_0 \xleftarrow{\psi} G_1\,,$$

we first compute a free resolution of M up to stage $i+1$:

$$0 \longleftarrow M \longleftarrow F_0 \longleftarrow F_1 \longleftarrow \cdots \longleftarrow F_i \longleftarrow F_{i+1}\,.$$

Then we consider the induced commutative diagram with exact rows and columns below:

$$\begin{array}{ccccc}
\mathrm{Hom}_R(F_{i+1},N) & \longleftarrow & \mathrm{Hom}_R(F_i,N) & \longleftarrow & \mathrm{Hom}_R(F_{i-1},N) \\
\uparrow & & \uparrow & & \uparrow \\
\mathrm{Hom}_R(F_{i+1},G_0) & \xleftarrow{\texttt{a2}} & \mathrm{Hom}_R(F_i,G_0) & \xleftarrow{\texttt{a1}} & \mathrm{Hom}_R(F_{i-1},G_0) \\
\uparrow{\scriptstyle \texttt{b2}} & & \uparrow{\scriptstyle \texttt{b1}} & & \\
\mathrm{Hom}_R(F_{i+1},G_1) & \longleftarrow & \mathrm{Hom}_R(F_i,G_1)\,. & &
\end{array}$$

In this diagram, the relevant maps for constructing $\mathrm{Ext}^i_R(M,N)$ are indicated. Here is the complete procedure:

```
proc Ext_Mod (int i, module phi,psi)
"USAGE:   Ext_Mod(i,phi,psi);    i int, phi,psi modules
RETURN:  module
NOTE:    The generators for the output module are the columns of a
         presentation matrix for Ext^i(coker(phi),coker(psi)).
EXAMPLE: example Ext_Mod; shows an example
"
{
  if (i<0) {return([1]);}
  def ResPhi = mres(phi,i+1);
  module M1,M2;
  M2 = ResPhi[i+1];
  if (i==0) {M1 = 0;} else {M1 = ResPhi[i];}
```

```
  int row = nrows(psi);
  module a1 = transpose( tensor( diag(1,row), M1 ) );
  module a2 = transpose( tensor( diag(1,row), M2 ) );
  module b1 = tensor( psi, diag(1,ncols(M1)) );
  module b2 = tensor( psi, diag(1,ncols(M2)) );
  module A = modulo(a2,b2);
  module B = modulo(A,a1+b1);
  return(B);
}
example
{ "EXAMPLE:"; echo = 2;
  ring R = 0, (x,y,z), dp;
  module phi,psi = [x2-y3],[x2-y5];
  print(Ext_Mod(0,phi,psi));
  print(Ext_Mod(1,phi,psi));
}
```

□

Exercise 3.5. To construct the first surface (and to check smoothness), proceed as for the Veronese surface in Example 4.13:

```
> ring S = 32003, x(0..4), dp;
> resolution kos = nres(maxideal(1),0);
> print(betti(kos),"betti");
> matrix alpha0 = random(32002,10,5);
> matrix pres = module(alpha0)+kos[4];
> matrix dir = transpose(pres);
> resolution fdir = mres(dir,2);
> print(betti(fdir),"betti");
> LIB "matrix.lib";
> ideal I = flatten(fdir[2]);
```

Now, compute the minimal free resolution and display the Betti diagram:

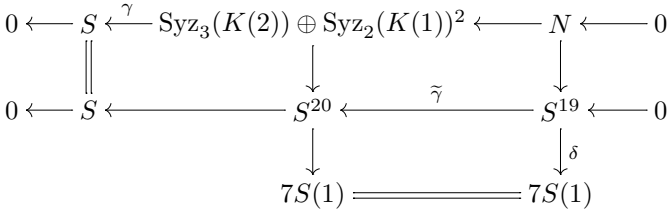
```
> resolution FI = mres(I,0);
> print(betti(FI),"betti");
```

Comparing with the Betti diagram obtained by resolving the ideal of the surface `QES` in Exercise 1.5, you will see that the two diagrams coincide. In fact, the two surfaces belong to the same family of smooth surfaces, they are quintic elliptic scrolls. See Decker, Ein, and Schreyer (1993).

The construction of the second surface is a bit more subtle. As in Example 4.14, every homomorphism

$$S \xleftarrow{\gamma} \mathrm{Syz}_3(K(2)) \oplus \mathrm{Syz}_2(K(1))^2$$

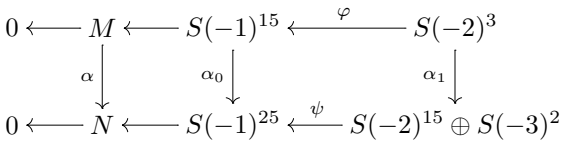
lifts to a homomorphism $S \longleftarrow S^{20}$ which fits into a commutative diagram with exact rows and columns:



To construct a "generic" γ and $N = \ker \gamma$, start by choosing $\widetilde{\gamma}$ at random:

```
> ring S1 = 32003, x(0..4), dp;
> resolution kos = nres(maxideal(1),0);
> betti(kos);
> matrix gammatilde = random(32002,20,19);
> matrix kos1 = matrix(kos[1]);
> matrix kos2 = kos[2];
> LIB"matrix.lib";
> matrix kos2pluskos1pluskos1 = dsum(kos2,kos1,kos1);
> module delta = kos2pluskos1pluskos1*gammatilde;
> attrib(delta,"isHomog",intvec(-1,-1,-1,-1,-1,-1,-1));
> resolution fdelta = mres(delta,0);
> print(betti(fdelta),"betti");
           0     1     2     3     4     5
-------------------------------------------
   -1:     7    19    25    15     3     -
    0:     -     -     -     2     3     1
-------------------------------------------
total:     7    19    25    17     6     1
```

From the free presentations of $M = \mathrm{Syz}_4(K(3))^3$ and N, it is clear how to construct the desired homomorphism α:



```
> matrix psi = matrix(fdelta[3]);
> matrix talpha1 = random(32002,3,15);
> matrix zero[3][2];
> talpha1 = concat(talpha1,zero);
> matrix kos5 = kos[5];
> matrix tphi = transpose(dsum(kos5,kos5,kos5));
> matrix talpha1tilde = talpha1*transpose(psi);
> matrix talpha0 = lift(tphi,talpha1tilde);
```

Next, construct the cokernel of α by taking a mapping cone:

```
> matrix dir = transpose(concat(psi,transpose(talpha0)));
> resolution fdir = mres(dir,2);
> print(betti(fdir),"betti");
> ideal I = groebner(flatten(fdir[2]));
> resolution FI = mres(I,0);
> print(betti(FI),"betti");
```

Finally, check smoothness as in Example 4.13. □

Exercise 4.1. Apply the command `primdecGTZ` from `primdec.lib` to compute a primary decomposition of the given ideal:

```
> ring R = 0, (t,w,x,y,z), dp;
> ideal I = w2xy+w2xz+w2z2, tx2y+x2yz+x2z2, twy2+ty2z+y2z2,
.           t2wx+t2wz+t2z2;
> LIB "primdec.lib";
> list L = primdecGTZ(I);
```

Next, compute the number of components, check whether the components are prime, and compute their dimension:

```
> int s = size(L);   // number of components
> int i,j;
> for (i=1; i<=s; i++)
. {
.   L[i][1]=std(L[i][1]);
.   L[i][2]=std(L[i][2]);
.   // test for primeness:
.   if ( size( reduce( lead(L[i][2]), std(lead(L[i][1])) ) ) > 0 )
.   {
.     print(string(i)+"th component is not prime and of dimension "
.       +string(dim(L[i][2]))+".");
.   }
.   else
.   {
.     print(string(i)+"th component is prime and of dimension "
.       +string(dim(L[i][2]))+".");
.   }
. }
```

Finally, check which of the components are embedded:

```
> for (i=1; i<=s; i++)
. {
.   for (j=1; j<=s; j++)
.   {
.     if (( dim(L[j][2]) > dim(L[i][2]) ) and
.        ( size(reduce(L[j][2],L[i][2],1)) == 0 ))
.     {
.       print(string(i)+"th component is embedded.");
```

```
.       j=s;
.     }
.   }
. }
```

□

Exercise 4.2. Apply the command `normal` from `normal.lib`:

```
> ring R = 0, (b,s,t,u,v,w,x,y,z), dp;
> ideal I =  wy-vz, vx-uy, tv-sw, su-bv, tuy-bvz;
> LIB "normal.lib";
> list nor = normal(I);
```

The result is a list of three rings. Follow the instructions displayed by `SINGULAR` to get information on these rings:

```
> for (int i=1; i<=size(nor); i++) { def R_nor(i) = nor[i]; }
> setring R_nor(1);
> R_nor(1); norid; normap;
> setring R_nor(2);
> R_nor(2); norid; normap;
> setring R_nor(3);
> R_nor(3); norid; normap;
```

Compare the result with the primary decomposition of I:

```
> setring R;
> primdecGTZ(I);
```

The output shows that I is the intersection of three prime ideals. In particular, it is radical. Each normalization map is an isomorphism onto the affine domain defined by the corresponding prime component. That is, these domains are already normal, and the normalization algorithm just separates the prime components of I. Geometrically, we get the linear subspaces $\mathrm{V}(u,v,w)$ and $\mathrm{V}(s,v,y)$ of $\mathbb{P}^8$, and the variety defined by the 2×2 minors of the "generic" 3×3 matrix with rows (b,s,t), (u,v,w), and (x,y,z). □

Exercise 4.3. Making use of elimination, compute a representation for the subalgebra of $\mathbb{F}_{32003}[x,y,z]/\langle z^2 - x^5 - y^5 \rangle$ described in the hint to the exercise as an affine ring S/J:

```
> ring R = 32003, (x,y,z,a,b,c,d), dp;
> ideal I = a-xy, b-y2, c-yz, d-y3;
> I = I, z2-x5-y5;
> ideal J = eliminate(I,y);
> ring S = 32003, (x,z,a,b,c,d), dp;
> ideal J = imap(R,J);
> attrib(J,"isSB",1);
> J;
```

To check that S/J has the desired properties, compute its dimension and its normalization:

```
> dim(J);
> LIB "normal.lib";
> list nor = normal(J);
> def R_nor = nor[1]; setring R_nor;
> norid; normap;
```
□

Exercise 4.4. (a) Proceed as follows:

```
> ring R = 0, x(1..3), dp;
> matrix D[3][3] = x(1), x(2), x(3)^2-1,
.                  x(2), x(3), x(1)*x(2)+x(3)+1,
.                  x(3)^2-1, x(1)*x(2)+x(3)+1, 0;
> ideal I = det(D), x(1)*x(3)-x(2)^2;
> matrix Df = jacob(I);
> I = std(I);
> int c = nvars(R) - dim(I); c;
```

The output shows that I has codimension 2. Since it is generated by 2 elements, it has pure codimension 2 and is unmixed (see Section 5.3 on Cohen-Macaulay rings). To verify this with `SINGULAR`, check that I coincides with its equidimensional hull:

```
> LIB "primdec.lib";
> ideal I_max = equidimMax(I);
> size(reduce(I_max,I));
```

(b) Define J and compute the saturation of I with respect to J:

```
> ideal J = minor(Df,c), I;
> J = groebner(J);
> sat(I,J);
```

The output shows that $I : J^2 = I : J^\infty = \mathbb{Q}[x_1, x_2, x_3]$. Hence, $J^2 \subset I \subset J$ and, thus, $\mathrm{V}(I) = \mathrm{V}(J)$. Of course, this could also be verified by checking that $\sqrt{I} = \sqrt{J}$.

(c) As for I, check that J coincides with its equidimensional hull.

```
> ideal J_max = equidimMax(J);
> size(reduce(J_max,J));
```

Thus, J is unmixed and has pure codimension 2. To show that $A = \mathrm{V}(J) \subset \mathbb{A}^3(\mathbb{C})$ is smooth, apply the Jacobian criterion:

```
> matrix DJ = jacob(J);
> ideal SLoc = minor(DJ,c), J;
> groebner(SLoc);
```

The output shows that the ideal `SLoc` contains 1. According to the Jacobian criterion, this implies in particular that J is a radical ideal. Of course, this could also be verified by checking that $\sqrt{J} = J$. □

Exercise 4.5. To write the desired procedure, we make use of the procedure `maximaldegree` from Exercise 1.4.

```
proc zeros (poly f)
"USAGE:   zeros(f);   f poly
ASSUME:  the active ring is univariate
RETURN:  ring
NOTE:    In the output ring, f decomposes into linear factors. The
         list of solutions of f=0 may be accessed by typing, for
         instance,
            def RRR=zeros(f); setring RRR; ZEROS;
"
{
  if (defined(primitive)==0){ LIB "primitiv.lib"; }
  if (nvars(basering)!=1)
    { ERROR("The active ring is not univariate"); }
  if (deg(f)<=0){ ERROR("f is a constant polynomial"); }
  def R_aux = basering;
  ideal facts_f = factorize(f,1);
  int d = maximaldegree(facts_f);
  int counter=size(facts_f);
  int i;
  while (d>1)
  {
    while (deg(facts_f[counter])<=1) { counter--; }
    if (defined(minpoly)) { poly g = minpoly; }
      else { poly g=0; }
    poly h = facts_f[counter];
    if (g==0)
    {
      ring R_aux2 = (char(basering),a), x, dp;
      map psi = R_aux,a;
      minpoly = number(psi(h));
      ideal facts_old = imap(R_aux,facts_f);
      ideal facts_f;
      for (i=1; i<=size(facts_old); i++)
      {
        facts_f = facts_f, factorize(facts_old[i],1);
      }
    }
    else
    {
      ring R_aux1 = char(basering), (a,x), dp;
      poly g = imap(R_aux,g);
      poly h = imap(R_aux,h);
      ideal facts_f = imap(R_aux,facts_f);
      ideal I = g,h;
      ideal Prim = primitive(I);
      poly MP_new = Prim[1];
```

```
      poly a_new = Prim[2];
      poly x_new = Prim[3];
      ring R_aux2 = (char(basering),a), x, dp;
      map psi = R_aux1,0,a;
      minpoly = number(psi(MP_new));
      poly a_new = psi(a_new);
      poly x_new = psi(x_new);
      map phi = R_aux1,a_new,x;
      ideal facts_old = phi(facts_f);
      ideal facts_f;
      for (i=1; i<=size(facts_old); i++)
      {
        facts_f = facts_f, factorize(facts_old[i],1);
      }
      kill R_aux1;
    }
    facts_f = simplify(facts_f,2);
    kill R_aux;
    def R_aux = R_aux2;
    setring R_aux;
    kill R_aux2;
    d = maximaldegree(facts_f);
  }
  ideal ZEROS = -subst(facts_f,x,0);
  export(ZEROS);
  return(R_aux);
}
```

We describe how to test the procedure by considering the third example given in the exercise:

```
> ring R = 167, x, dp;
> def RRR = zeros(x100-13);
> setring RRR;
> ZEROS;
> poly g = 1;
> for (int i=1; i<=size(ZEROS); i++) { g = g*(x-ZEROS[i]); }
> g;
```

□

Exercise 4.6. To begin with, compute the numerator D_0:

```
> ring R = 0, (x,y,z), dp;
> ideal MI_2 = maxideal(2);
> ideal MI_3 = maxideal(3);
> ideal MI_4 = maxideal(4);
> ring R_ext = 0, (a(1..6),b(1..6),c(1..6),x,y,z), dp;
> ideal MI_2 = imap(R,MI_2);
> matrix A[6][1] = a(1..6);
> matrix B[6][1] = b(1..6);
> matrix C[6][1] = c(1..6);
```

```
> poly F(0) = (matrix(MI_2)*A)[1,1];
> poly F(1) = (matrix(MI_2)*B)[1,1];
> poly F(2) = (matrix(MI_2)*C)[1,1];
> ideal G = x2*F(0), xy*F(0), xz*F(0), yz*F(0), x2*F(1),
.           xy*F(1), xz*F(1), y2*F(1), yz*F(1), x2*F(2),
.           xy*F(2), xz*F(2), y2*F(2), yz*F(2), z2*F(2);
> ideal MI_4 = imap(R,MI_4);
> matrix M = coeffs(G,MI_4,xyz);
> poly D_0 = det(M);
> deg(D_0); size(D_0);
```

The output shows that D_0 has degree 15 and consists of 37490 terms. Next, compute the extraneous factor D_0^{ext}:

```
> ideal G_ext = x2*F(1), x2*F(2), y2*F(2);
> ideal MI_ext = x2y2, x2z2, y2z2;
> matrix M_ext = coeffs(G_ext,MI_ext,xyz);
> poly D_0_ext = det(M_ext);
```

Entering the two lines below, you will define the resultant $\mathrm{Res}_{2,2,2}$ (up to sign), and you will see that it is a polynomial of degree 12 with 21894 terms:

```
> poly Res = D_0/D_0_ext;
> deg(Res); size(Res);
```

□

Exercise 4.7. Continuing the SINGULAR session above, compute the value of the resultant $\mathrm{Res}_{2,2,2}$ for the polynomials $F_0 = \frac{\partial F}{\partial x}$, $F_1 = \frac{\partial F}{\partial y}$, $F_2 = \frac{\partial F}{\partial z}$, where $F = F_a$:

$$F = a_1x^3 + a_2x^2y + a_3xy^2 + a_4y^3 + a_5x^2z + a_6xyz \\ +a_7y^2z + a_8xz^2 + a_9yz^2 + a_{10}z^3 \ :$$

```
> ring S = 0,(x,y,z,a(1..10)), (dp(2),dp(11));
> ideal MI_2 = imap(R,MI_2);
> ideal MI_3 = imap(R,MI_3);
> matrix A[10][1] = a(10),a(9),a(7),a(4),a(8),a(6),a(3),a(5),a(2),
.                   a(1);
> poly F = (matrix(MI_3)*A)[1,1];
> ideal J = diff(F,x), diff(F,y), diff(F,z);
> LIB "matrix.lib";
> map phi = R_ext, flatten(coeffs(diff(F,x),MI_2,xyz)),
.                  flatten(coeffs(diff(F,y),MI_2,xyz)),
.                  flatten(coeffs(diff(F,z),MI_2,xyz));
> poly D = phi(Res);
> deg(D); size(D);
```

As in Exercise 2.3, we get a polynomial D in $a_1, \ldots, a_{10}$ of degree 12 with 2040 terms. This polynomial coincides with that of Exercise 2.3 up to the constant factor -27. Note that **Euler's formula**

$$\deg(F) \cdot F = x \cdot \frac{\partial F}{\partial x} + y \cdot \frac{\partial F}{\partial y} + z \cdot \frac{\partial F}{\partial z}$$

implies that F vanishes at a point of $\mathbb{P}^2$ if all its first partial derivatives vanish. Hence, D defines the locus of points in $\mathbb{P}^9$ for which the plane curve $\mathrm{V}(F) \subset \mathbb{P}^2$ is singular (see the solution to Exercise 2.3). □

Exercise 5.1. (a) The solution requires to compute algebra relations and is, thus, based on elimination. More precisely, we make use of the following observation. Let $f_1, \ldots, f_s \in K[\boldsymbol{x}]$ be homogeneous polynomials which are sorted such that $\deg(f_1) \geq \ldots \geq \deg(f_s) \geq 1$. Write $L_i = \{1, \ldots, s\} \setminus \{i\}$. Suppose that for some j, we have $K[f_\nu \mid \nu \in L_i] \subsetneq K[f_1, \ldots, f_s]$, for all $i < j$. Then $K[f_\nu \mid \nu \in L_j] = K[f_1, \ldots, f_s]$ iff the ideal $J := \langle y_1 - f_1, \ldots, y_s - f_s \rangle \subset K[\boldsymbol{x}, \boldsymbol{y}]$ contains an element of type $y_j - g$, where $g \in K[y_{j+1}, \ldots, y_s]$.

```
proc min_generating_set (matrix P,S)
"USAGE:  min_generating_set(P,S);   P,S matrix
ASSUME: The entries of P,S are homogeneous and ordered by ascending
        degrees. The first entry of S equals 1. (As satisfied by
        the first two output matrices of invariant_ring(G).)
RETURN: ideal
NOTE:   The given generators for the output ideal form a minimal
        generating set for the ring generated by the entries of
        P,S. The generators are homogeneous and ordered by
        descending degrees.
"
{
  if (defined(flatten)==0) { LIB "matrix.lib"; }
  ideal I1,I2 = flatten(P),flatten(S);
  int i1,i2 = size(I1),size(I2);
  // We order the generators by descending degrees
  // (the first generator 1 of I2 is omitted):
  int i,j,s = i1,i2,i1+i2-1;
  ideal I;
  for (int k=1; k<=s; k++)
  {
    if (i==0) { I[k]=I2[j]; j--; }
    else
    {
      if (j==0) { I[k]=I1[i]; i--; }
      else
      {
        if (deg(I1[i])>deg(I2[j])) { I[k]=I1[i]; i--; }
        else { I[k]=I2[j]; j--; }
      }
    }
  }
  intvec deg_I = deg(I[1..s]);
  int n = nvars(basering);
```

```
  def BR = basering;

  // Create a new ring with elimination order:
  //---------------------------------------------------------------
  // ****    this part uses the command ringlist which is      ****
  // ****     only available in SINGULAR-3-0-0 or newer        ****
  //---------------------------------------------------------------
  list rData = ringlist(BR);
  intvec wDp;
  for (k=1; k<=n; k++) {
    rData[2][k] ="x("+string(k)+ ")";
    wDp[k]=1;
  }
  for (k=1; k<=s; k++) { rData[2][n+k] ="y("+string(k)+ ")"; }
  rData[3][1] = list("dp",wDp);
  rData[3][2] = list("wp",deg_I);
  def R_aux = ring(rData);
  setring R_aux;
  //---------------------------------------------------------------
  ideal J;
  map phi = BR, x(1..n);
  ideal I = phi(I);
  for (k=1; k<=s; k++) { J[k] = y(k)-I[k]; }
  option(redSB);
  J = std(J);

  // Remove all generators that are depending on some x(i) from J:
  int s_J = size(J);
  for (k=1; k<=s_J; k++) { if (J[k]>=x(n)) {J[k]=0;} }

  // The monomial order on K[y] is chosen such that linear leading
  // terms in J are in 1-1 correspondence to superfluous generators
  // in I :
  ideal J_1jet = std(jet(lead(J),1));
  intvec to_remove;
  i=1;
  for (k=1; k<=s; k++)
  {
    if (reduce(y(k),J_1jet)==0){ to_remove[i]=k; i++; }
  }
  setring BR;
  if (to_remove == 0) { return(ideal(I)); }
  for (i=1; i<=size(to_remove); i++)
  {
    I[to_remove[i]] = 0;
  }
  I = simplify(I,2);
  return(I);
}
```

In versions of SINGULAR prior to 3-0-0, creating a new ring in a procedure as above typically required the use of the execute[1] command:

```
// Create a new ring with elimination order:
//--------------------------------------------------------------
if (minpoly<>0){ string @MP=string(minpoly); }
string newring =
   "ring R_aux = (" + charstr(BR) + "),(x(1.." + string(n)
    + "),y(1.." + string(s) + ")),(dp(" + string(n) + "),wp("
    + string(deg_I) + "));";
execute(newring);
if (defined(@MP)) { @MP = "minpoly=" + @MP; execute(@MP); }
//--------------------------------------------------------------
```

To test the procedure, we apply it to an example with superfluous generators:

```
> ring R1 = 0, (x,y), dp;
> matrix P[1][3] = x2+y2, x2-y2, x3-y3;
> matrix S[1][5] = 1, x-y, x3-xy2, x4-y4, xy3+y4;
> min_generating_set(P,S);
```

The output shows that x^2-y^2, x^2+y^2, and $x-y$ form a minimal set of generators.

(b) We apply the procedure from part (a) to Example 8.9. It turns out that the ring of invariants under consideration has a fundamental generator of degree 5. Thus, Noether's degree bound does not hold in this case (the group order is 4).

```
> ring R = 2, x(1..4), dp;
> matrix A[4][4];
> A[1,4]=1; A[2,1]=1; A[3,2]=1; A[4,3]=1;
> LIB "finvar.lib";
> matrix P,S = invariant_ring(A);
> ideal MGS = min_generating_set(P,S);
> deg(MGS[1]);
```

□

Exercise 5.2. (a)

```
proc is_unit (poly f)
"USAGE:  is_unit(f);   f poly
RETURN: int;  1 if f is a unit in the active ring,
              0 otherwise.
"
{
  return(leadmonom(f)==1);
}
```

[1] The execute command may give rise to name conflicts. Moreover, procedures which make use of the execute command cannot be precompiled (a feature which future versions of SINGULAR will provide). Therefore, we recommend to use the execute command only if it is really needed.

We apply the procedure to the polynomial $3 + x$, regarded as an element of four different rings. These rings are implemented by orders which are global, local, and mixed, respectively:

```
> ring R = 0, (x,y), dp;
> poly f = 3+x;
> is_unit(f);
> ring R1 = 0, (x,y), ds;
> is_unit(imap(R,f));
> ring R2 = 0, (x,y), (ds(1),dp);
> is_unit(imap(R,f));
> ring R3 = 0, (x,y), (dp(1),ds);
> is_unit(imap(R,f));
```

(b) Let $u_0 \in K \setminus \{0\}$, and let $u_1 \in K[\boldsymbol{x}]$, with $u_1(0) = 0$. Then

$$(u_0 - u_0 u_1)^{-1} = \frac{1}{u_0} \cdot \left(1 + \sum_{k=1}^{\infty} u_1^k\right) \in K[[\boldsymbol{x}]]\,.$$

This formula is used by the following procedure.

```
proc invert_unit (poly u, int d)
"USAGE:   invert_unit(u,d);   u poly, d int
RETURN:  poly;
NOTE:    If u is a unit in the active ring, the output polynomial
         is the power series expansion of the inverse of u up to
         order d. Otherwise, the zero polynomial is returned.
"
{
  if (is_unit(u)==0) { return(poly(0)); }
  poly u_0 = jet(u,0);
  u = jet(1-u/u_0,d);
  poly u_1 = u;
  poly inv = 1 + u_1;
  for (int i=2; i<=d; i++)
  {
    u_1 = jet(u_1*u,d);
    inv = inv + u_1;
  }
  return(inv/u_0);
}
```

□

Exercise 5.3. We start by computing the singular locus of the affine cone over C and the minimal associated primes of the resulting ideal:

```
> LIB "primdec.lib";
> ring R = 0, (x,y,z), dp;
> poly f = ((x4+y4-z4)^4-x2y5z9)*(x4+y4-z4);
> ideal Slocf = f,jacob(f);
> list SLoc = minAssGTZ(Slocf);
> SLoc;
```

From the output, we see that the singular locus consists of the four rational points $(0:1:1)$, $(0:-1:1)$, $(1:0:1)$, $(-1:0:1)$, two pairs of points which are conjugate over $\mathbb{Q}$ (defined by the primes $\langle x, y^2+z^2\rangle$ and $\langle y, x^2+z^2\rangle$), and a quadruple of pairwise conjugate points over $\mathbb{Q}$ (defined by the prime $\langle z, x^4+y^4\rangle$). We describe the branches of C at some of these points. To begin with, consider $p=(0:1:1)$:

```
> LIB "hnoether.lib";
> ring R_loc1 = 0, (u,v), ds;
> map phi = R,u,v-1,1;
> poly f = phi(f);
> def L1 = hnexpansion(f);
> def HNE_ring1 = L1[1];
> setring HNE_ring1;
> list INV = invariants(hne);
> // Number of branches:
. size(INV)-1;
> // Intersection Multiplicities of the branches:
. print(INV[size(INV)][2]);
```

The output shows that there are three branches. Two of the branches intersect each other with multiplicity two and intersect the third branch transversally. In particular, all branches are smooth. Smoothness can also be seen by checking the invariants of the branches. For instance, a branch is smooth iff its δ-invariant is zero:

```
> for (int i=1; i<size(INV); i++)
. {
.   if (INV[i][5]==0){ print("branch No."+string(i)+" is smooth");}
. }
```

The same picture is obtained for a point defined by the prime $\langle x, y^2+z^2\rangle$. To see this, first enter the code below. Then, proceed as above.

```
> ring R_loc2 = (0,a), (u,v), ds;
> minpoly = a2+1;
> map phi = R,u,v-a,1;
> poly f = phi(f);
> def L2 = hnexpansion(f);
> def HNE_ring2 = L2[1];
> setring HNE_ring2;
> displayInvariants(hne);
```

Finally, consider a point corresponding to the prime $\langle z, x^4+y^4\rangle$:

```
> ring R_loc3 = (0,a), (u,v), ds;
> minpoly = a4+1;
> map phi = R,1,v-a,u;
> poly f = phi(f);
> def L3 = hnexpansion(f);
> displayInvariants(L3);
```

Checking, for instance, the δ-invariant, we see that there is one smooth and one singular branch. The branches intersect each other with multiplicity 9. □

Exercise 5.4. (a) Determine the singular locus of the affine cone over C:

```
> ring R = 0, (x,y,z), dp;
> poly f = 3y3-3xy2-2xy3+x2y3+x3;
> poly C = homog(f,z);
> ideal I = jacob(C);
> I = std(I);
> LIB "primdec.lib";
> list SLoc = primdecGTZ(I);
> SLoc;
```

The output shows that C has exactly 4 singular points:

$$(1:1:1),\ (0:1:0),\ (1:0:0),\ (0:0:1)$$

(in addition to the primary components defining these points, the ideal `SLoc` has an embedded component corresponding to the origin of $\mathbb{A}^3$). We already see that the first two singular points are nodes (for each point, the corresponding primary component coincides with its radical). Further, the point $(1:0:0)$ is a simple cusp (the Tjurina number at this point is $\dim_{\mathbb{C}} \mathbb{C}\{y,z\}/\langle y^2, z\rangle = 2$). For the singular point $(0:0:1)$, the 2-jet of the given equation is zero, while the 3-jet is nonzero. To show that the singularity at $(0:0:1)$ is an ordinary 3-multiple point, it, thus, remains to check that the 3-jet decomposes into 3 different linear factors over $\mathbb{C}$:

```
> factorize(jet(f,3));
```

The output shows that the 3-jet is irreducible over $\mathbb{Q}$. Thus, it cannot have a multiple factor over $\mathbb{C}$.

(b) Proceed as follows:

```
> ideal Adj_S = 1;
> for (int k=1; k<=3; k++)
. {
.   Adj_S = intersect(Adj_S,SLoc[k][2]);
. }
> Adj_S = intersect(Adj_S,SLoc[k][2]^2);
> ideal Adj_LS_3 = jet(std(Adj_S),3);
> Adj_LS_3 = simplify(Adj_LS_3,6);  Adj_LS_3;
```

From the output, we see that the four polynomials $xyz-y^2z,\ x^2z-y^2z,\ xy^2-y^2z$, and x^2y-y^2z span $\mathcal{L}_3$. We randomly choose two $\mathbb{Q}$-linear combinations f_1, f_2 of these polynomials, and compute the sets B_1 and B_2 of intersection points of $\mathrm{V}(f_1)$ and $\mathrm{V}(f_2)$ with C outside the singular locus of C:

```
> LIB "random.lib";
> def f(1),f(2) = randomid(Adj_LS_3,2,10);
> ideal I(1) = f(1),C;
> ideal I(2) = f(2),C;
> ideal B(1) = sat(I(1),I)[1];
> ideal B(2) = sat(I(2),I)[1];
```

(c) Proceed as follows:

```
> Adj_S = intersect(Adj_S,B(1),B(2));
> option(redSB);
> ideal L' = jet(std(Adj_S),4);
> L' = simplify(L',6);
> poly f' =  randomid(L',1,10)[1];
> ideal I' = f',C;
> ideal B(3) = sat(I',L')[1];
```

(d) Proceed as follows:

```
> ideal L'' = jet(std(intersect(Adj_LS_3,B(3))),3);
> L'' = simplify(L'',6);  L'';
```

From the output, we read that $\mathcal{L}''$ is a one-dimensional linear system. We take f_1'', f_2'' to be the given generators:

```
> poly f''(1),f''(2) = L'';
```

(e) Proceed as follows:

```
> ring R_t = (0,t), (x,y,z), dp;
> poly f'' = imap(R,f''(1)) + t*imap(R,f''(2));
> ideal I_t = f'', imap(R,C);
> I_t = std(I_t);
> ideal L'' = imap(R,L'');
> I_t = sat(I_t,L'')[1];
> I_t = std(subst(I_t,z,1));
> def phi_x = reduce(x,I_t);  phi_x;
> def phi_y = reduce(y,I_t);  phi_y;
```

To double check that the map φ with components φ_x and φ_y is indeed a parametrization of C, we verify that the image of φ is contained in C, and compute the Zariski closure of the image.

```
> map testmap = R, phi_x, phi_y, 1;
> testmap(C);
> ring S = 0, (t,x,y), dp;
> ideal I_t = imap(R_t,I_t);
> eliminate(I_t,t);
```

□

References

Abo, H.; Decker, W.; Sasakura, N. (1998): An elliptic conic bundle in $\mathbb{P}^4$ arising from a stable rank–3 vector bundle. *Math. Z.* **229**, 725–741.

Adams, W.W.; Loustaunau, P. (1994): *An introduction to Gröbner bases.* Graduate Studies in Mathematics, 3. AMS, Providence, RI.

Amrhein, B.; Gloor, O. (1998): The Fractal Walk. In: B. Buchberger and F. Winkler (eds.): *Gröbner Bases and Applications,* 305–322, LNS 251, CUP, Cambridge.

Amrhein, B.; Gloor, O.; Küchlin, W. (1997): On the Walk. *Theoret. Comp. Sci.* **187**, 179–202.

Apel, J. (1988): *Gröbnerbasen in nichtkommutativen Algebren und ihre Anwendung.* Dissertation, Universität Leipzig.

Arbarello, E., Cornalba, M., Griffiths, P.A., Harris, J. (1985): *Geometry of Algebraic Curves.* Springer-Verlag.

Arnold, V.I.; Gusein-Zade, S.M.; Varchenko, A.N. (1985): *Singularities of differentiable maps.* Birkhäuser.

Artin, M. (1976): *Deformations of singularities.* Tata Institute of Fundamental Research, Bombay.

Atiyah, M.F.; McDonald, I.G. (1969): *Introduction to commutative algebra.* Addison-Wesley.

Aubry, P.; Lazard, D.; Moreno Maza, M. (1999): On the Theories of Triangular Sets. *J. Symb. Comput.* **28**, 105–124.

Aubry, P.; Moreno Maza, M. (1999): Triangular Sets for Solving Polynomial Systems: a Comparitive Implementation of Four Methods. *J. Symb. Comput.* **28**, 125–154.

Aure, A.B.; Decker, W.; Hulek, K.; Popescu, S.; Ranestad, K. (1997): Syzygies of abelian and bielliptic surfaces in $\mathbb{P}^4$. *International J. Math.* **8** , 849–919.

Avramov, L.L..; Grayson, D.R. (2002): Resolutions and Cohomology over Complete Intersections. In: Eisenbud et al (2002), 215–249.

Babbage, C. (1836): *Scribbling books*, volume 2, Science Museum Library, London.

Barth, W.; Hulek, K.; Moore, R. (1987): Degenerations of Horrocks-Mumford surfaces. *Math. Ann.* **277**, 735–755.

Barth, W.; Hulek, K.; J.; Peters, C.A.M.; Van de Ven, A. (2004): *Compact Complex Surfaces.* Second enlarged edition. Springer-Verlag.

Bayer, D.; Stillman, M. (1987): A theorem on refining division orders by the reverse lexicographic orders. *Duke J. Math.* **55**, 321–328.

Besche, H.U.; Eick, B.; O'Brien, E.A. (2001): The groups of order at most 2000. *Electron. Res. Announc. Amer. Math. Soc.* **7**, 1–4.

Berlekamp, E.R. (1967): Factoring polynomials over finite fields. *Bell System Tech. J.* **46**, 1853–1859.

Berlekamp, E.R. (1970): Factoring polynomials over large finite fields. *Math. Comp.* **24**, 713–735.

Bernstein, D.N. (1975): The number of roots of a system of equations. *Funct. Anal. Appl.* **9**, 183–185; translation from *Funkts. Anal. Prilozh.* **9**, No. 3, 1–4.

Bernstein, I.N.; Gelfand, I.M.; Gelfand, S.I. (1978): Algebraic vector bundles on $\mathbb{P}^n$ and problems of linear algebra. *Functional Anal. Appl.* **12**, 212–214.

Böhm, J. (1999): *Parametrisierung rationaler Kurven.* Diplomarbeit, Universität Bayreuth.

Brickenstein, M. (2004): *Neue Varianten zur Berechnung von Gröbnerbasen.* Diplomarbeit, TU Kaiserslautern.

Brieskorn, E.; Knörrer, H. (1986): *Plane algebraic curves.* Birkhäuser.

Bruns, W.; Herzog, J. (1993): *Cohen-Macaulay rings.* Cambridge Univ. Press.

Buchberger, B. (1965): *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenrings.* Dissertation, Universität Innsbruck.

Buchberger, B. (1970): Ein algorithmisches Kriterium für die Lösbarkeit eines algebraischen Gleichungssystems. *Aequationes mathematicae* **4**, 374–383.

Campillo, A. (1980): *Algebroid curves in positive characteristic.* SLN 813, Springer-Verlag.

Canny, J.F.; Emiris, I.Z. (1993): An efficient algorithm for the sparse mixed resultant. *AAECC*, 89–104.

Canny, J.F.; Emiris, I.Z. (1995): Efficient incremental algorithms for the sparse resultant and the mixed volume. *J. Symb. Comput.* **20**, 117–149.

Canny, J.F.; Emiris, I.Z. (2000): A subdivision-based algorithm for the sparse resultant. *J. ACM* **47** (3), 417–451.

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

Cattani, E.; Dickenstein, A. (2005): Introduction to residues and resultants. In: Dickenstein and Emiris eds. (2005), 1–62.

Cayley, A. (1889): *The Collected Mathematical Papers.* Cambridge University Press, Cambridge.

Chèze, G.; Lecerf, G. (2005): Lifting and recombination techniques for absolute factorization. *Manuscript, Université de Versailles*, Saint-Quentin-en-Yvelines.

Cohen, I.S. (1946): On the structure and ideal theory of complete local rings. *Trans. Amer. Math. Soc.* **59**, 252–261.

Collart, S.; Kalkbrener, M.; Mall, D. (1997): Converting Bases with the Gröbner Walk. *J. Symb. Comput.* **24**, 465–470.

Cox, D.; Little, J.; O'Shea, D. (1997): *Ideals, Varieties, and Algorithms.* 2nd edition, Springer-Verlag.

Cox, D.; Little, J.; O'Shea, D. (1998): *Using Algebraic Geometry.* Springer-Verlag.

Czapor, S.R. (1989): Solving algebraic equations: Combining Buchberger's algorithm with multivariate factorization. *J. Symb. Comput.* **7**(1), 49–53.

Decker, W. (1984): Das Horrocks-Mumford-Bündel und das Modul-Schema für stabile 2-Vektorbündel über $\mathbb{P}_4$ mit $c_1 = -1$, $c_2 = 4$. *Math. Z.* **188**, 101–110.

Decker, W.; Ein, L.; Schreyer, F.-O. (1993): Construction of surfaces in $\mathbb{P}^4$. *J. Alg. Geom.* **2**, 185–237.

Decker, W.; Eisenbud, D. (2002): Sheaf Algorithms Using the Exterior Algebra. In: Eisenbud et al (2002), 215–249.

Decker, W., Greuel, G.-M., Pfister, G. (1999): Primary decomposition: algorithms and comparisons, In: B.H. Matzat et al (eds.), *Algorithmic algebra and number theory, Heidelberg 1997*, 187–220, Springer-Verlag.

Decker, W.; Heydtmann, A.; Schreyer, F.-O. (1998): Generating a noetherian normalization of the invariant ring of a finite group. *J. Symb. Comput.* **25**, 727–731.

Decker, W.; de Jong, T. (1998): Gröbner bases and invariant theory. In: B. Buchberger and F. Winkler (eds.): *Gröbner Bases and Applications,* 305–322, LNS 251, CUP, Cambridge.

Decker, W.; de Jong, T.; Greuel, G.-M.; Pfister, G. (1999): The normalization: a new algorithm, implementation and comparisons. In: P. Draexler et al (eds.), *Computational methods for representations of groups and algebras. Proceedings of the Euroconference in Essen, Germany, April 1-5, 1997*, 267–285. Birkhäuser, Basel.

Decker, W.; Schreyer, F.O. (2000): Non-general type surfaces in $\mathbb{P}^4$: some remarks on bounds and constructions. *J. Symb. Comput.* **29**, 545–582.

Decker, W.; Schreyer, F.O. (2001): Computational algebraic geometry today. In: C. Ciliberto et al (eds.): *Application of Algebraic Geometry to Coding Theory, Physics, and Computation,* 65–120, Kluwer.

Decker, W.; Schreyer, F.O. (2006): *Varieties, Gröbner Bases, and Algebraic Curves.* To appear.

Derksen, H. (1999): Computation of invariants for reductive groups. *Adv. Math.* **141**, 366–384.

Derksen, H.; Kemper, G. (2002): *Computational invariant theory.* Springer-Verlag.

Dickenstein, A.; Emiris, I.Z. eds. (2005): *Solving Polynomial Equations. Foundations, Algorithms, and Applications.* Springer-Verlag.

Dolgachev, I. (1982): Weighted projective varieties. In: *Group actions and vector fields.* Proc. Pol.-North Am. Semin., Vancouver 1981, Lect. Notes Math. 956, 34–71, Springer-Verlag.

Eisenbud, D. (1995): *Commutative algebra with a view toward algebraic geometry.* Springer-Verlag.

Eisenbud, D. (1998): Computing cohomology. A chapter in W. Vasconcelos, *Computational methods in commutative algebra and algebraic geometry.* Springer-Verlag.

Eisenbud, D. (2005): *The geometry of Syzygies.* Springer-Verlag.

Eisenbud, D.; Fløystad, G.; Schreyer, F.O. (2003): Sheaf cohomology and free resolutions over exterior algebras. *Trans. Amer. Math. Soc.* **355**, no. 11, 4397–4426.

Eisenbud, D.; Grayson, D.R.; Stillman M.; Sturmfels, B. eds. (2002): *Computations in Algebraic Geometry with Macaulay 2.* Springer-Verlag.

Eisenbud, D.; Harris, J. (2000): *The geometry of schemes.* Graduate Texts in Mathematics, 197. Springer-Verlag.

Eisenbud, D.; Huneke, C.; Vasconcelos, W. (1992): Direct methods for primary decomposition. *Invent. Math.* **110**, 207–235.

Eisenbud, D.; Popescu, S. (1999): Gale duality and free resolutions of ideals of points. *Invent. Math.* **136**, 419–449.

Eisenbud, D.; Popescu, S. (1999): The projective geometry of the Gale transform. *J. Algebra* **230**, 127–173.

Eisenbud, D.; Schreyer, F.O. (2003): Resultants and Chow forms via exterior syzygies. *J. Amer. Math. Soc.* **16**, 537–579.

Faugère, C.; Gianni, P.; Lazard, D.; Mora, T. (1993): Efficient Computation of Zero–dimensional Gröbner Bases by Change of Ordering. *J. Symb. Comput.* **16**, 329–344.

Frühbis-Krüger, A.; Krüger, K.; Schönemann, H. (2003): Dynamic Modules in `SINGULAR`. *Reports on Computer Algebra* **32**, ZCA, TU Kaiserslautern. Online available at `http://www.mathematik.uni-kl.de/~zca`.

Gao, S. (2003): Factoring multivariate polynomials via partial differential equations. *Math. Comp.*, **72**, 801–822.

von zur Gathen, J.; Gerhard, J. (1999): *Modern computer algebra.* Cambridge Univ. Press.

Geddes, K.; Czapor, S.R.; Labahn, G. (1992): *Algorithms for computer algebra.* Kluwer Academic Publishers, Boston.

Gelfand, I.M.; Kapranov, M.M.; Zelevinsky, A.V. (1994): *Discriminants, resultants and multidimensional determinants.* Birkhäuser Verlag.

Gianni, P.; Trager, B.; Zacharias, G. (1988): Gröbner bases and primary decomposition of polynomial ideals. *J. Symb. Comput.* **6**, 149–167.

Giovini, A.; Mora, T.; Niesi, G.; Robbiano, L.; Traverso, C. (1991): One sugar cube, please, or selection strategies in the Buchberger algorithm. In S.M. Watt, editor, *Proc. ISSAC'91*, ACM Press, 49–54.

Gordan, P. (1899): Neuer Beweis des Hilbertschen Satzes über homogene Funktionen. *Nachrichten König. Ges. der Wiss. zu Gött.*, 240–242

Gräbe, H.-G. (1995a): On Factorized Gröbner Bases. In: J. Fleischer et al (eds.), *Computer Algebra in Science and Engineering*. World Scientific Singapore, 77–89.

Gräbe, H.-G. (1995b): Triangular Systems and Factorized Gröbner Bases. In: *Proceedings AAECC Paris, LNCS* **948**, Springer-Verlag, 248–261.

Grauert, H. (1972): Über die Deformation isolierter Singularitäten analytischer Mengen. *Invent. Math.* **15**, 171–198.

Grauert, H.; Remmert, R. (1971): *Analytische Stellenalgebren.* Springer-Verlag.

Greuel, G.-M.; Pfister, G. (2002): *A `SINGULAR` introduction to commutative algebra.* Springer-Verlag.

Greuel, G.-M.; Lossen, C.; Shustin, E. (2006): *Introduction to Singularities and Deformations.* To appear.

Gusein-Zade, S.M.; Nekhoroshev, N.N. (2000): Singularities of type A_k on plane curves of a chosen degree. *Funct. Anal. Appl.* **34**, No. 3, 214–215.

Harris, J. (1992): *Algebraic Geometry.* Springer-Verlag.

Hartshorne, R. (1977): *Algebraic Geometry.* Springer-Verlag.

Hermann, G. (1926): Die Frage der endlich vielen Schritte in der Theorie der Polynomideale. *Math. Ann.* **95**, 736–788.

Heydtmann, A.E.: *Generating invariant rings of finite groups.* Diplomarbeit, Universität des Saarlandes, Saarbrücken (1999).

Hilbert, D. (1890): Über die Theorie der algebraischen Formen. *Math. Ann.* **36**, 473–534.

Hilbert, D. (1893): Über die vollen Invariantensysteme. *Math. Ann.* **42**, 313–373.

Hillebrand, D. (1999): *Triangulierung nulldimensionaler Ideale – Implementierung und Vergleich zweier Algorithmen.* Diplomarbeit, Universität Dortmund.

Hilton, P.; Stammbach, U. (1977): *A course in Homological Algebra.* Springer-Verlag.

Hironaka, H. (1964): Resolution of singularities of an algebraic variety over a field of characteristic zero. *Annals of Math.* **79**. I: 109–203; II: 205–326.

Hochster, M.; Roberts, J. (1974): Rings of Invariants of Reductive Groups Acting on Regular Rings are Cohen-Macaulay. *Adv. in Math.* **13**, 115–175.

Horrocks, G.; Mumford, D. (1973): A rank 2 vector bundle on $\mathbf{P}^4$ with 15,000 symmetries. *Topology* **12**, 63–81.

de Jong, T. (1998): An algorithm for computing the integral closure. *J. Symb. Comput.* **26**, 273–277.

de Jong, T., Pfister, G. (2000): *Local analytic geometry.* Vieweg.

Kahrimanian, H.G. (1953): *Analytical differentiation by a digital computer.* Master Thesis, Temple University, Philadelphia.

Kaltofen, E. (1982): Polynomial factorization. In: B. Buchberger et al (eds.), *Computer algebra*, 95–113, Springer-Verlag.

Kaltofen, E. (1990): Polynomial factorization 1982-1986. In: I. Simon (ed.), *Computers in mathematics*, 285–309. Marcel Dekker, New York.

Kaltofen, E. (1992): Polynomial factorization 1987-1991. In: D.V. Chudnovsky and R.D. Jenks (eds.), *Proceedings of LATIN'92, Sao Paulo*, 294–313. Springer-Verlag.

Kaltofen, E. (2003): Polynomial factorization: a success story. In: J.R. Sendra (ed.), *Proc. ISSAC'03, Philadelphia.* ACM Press, 3–4.

Kemper, G. (1996): Calculating invariant rings of finite groups over arbitrary fields. *J. Symb. Comput.* **21**, 351–366.

Kemper, G. (1999): An algorithm to calculate optimal homogeneous systems of parameters. *J. Symb. Comput.* **27**, 171–184.

Kemper, G. (2002): The Calculation of Radical Ideals in Positive Characteristic. *J. Symb. Comput.* **23**, 229–238.

Kemper, G.; Steel, A. (1999): Some algorithms in invariant theory of finite groups. In: P. Draexler et al (eds.), *Computational methods for representations of groups and algebras. Proceedings of the Euroconference in Essen, Germany, April 1–5, 1997*, 267–285. Birkhäuser, Basel.

Kreuzer, M.; Robbiano, L. (2000): *Computational Commutative Algebra 1.* Springer-Verlag.

Krick, T.; Logar, A. (1991): An algorithm for the computation of the radical of an ideal in the ring of polynomials. In: H.F. Mattson et al (eds.), *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, 9th International Symposium, AAECC-9, New Orleans, LA, USA, October 7-11, 1991, Proceedings*, 195–205, LNCS 539, Springer-Verlag.

La Scala, R.; Stillman, M. (1998): Strategies for computing minimal free resolutions. *J. Symb. Comput.* **26**, No. 4, 409–431.

Larcombe, P.J. (1999): On Lovelace, Babbage and the Origins of Computer Algebra. In: Wester (1999), 323–332.

Lazard, D. (1991): A new method for solving algebraic systems of positive dimension. *Discr. Appl. Math.* **33**, 147–160.

Lazard, D. (1992): Solving Zero-dimensional Algebraic Systems. *J. Symb. Comput.* **13**, 117–131.

Levandovskyy, V. (2005): *Non-commutative Computer Algebra for polynomial algebras: Gröbner bases, applications and implementation.* Dissertation, TU Kaiserslautern.

Li, H. (2002): *Noncommutative Gröbner bases and filtered-graded transfer.* Springer, 2002.

Looijenga, E. (1984): *Isolated singular points on complete intersections.* London Math. Soc. Lect. Notes, Vol. 77, Cambridge Univ. Press.

Macaulay, F. (1903): Some formulae in elimination. *Proc. London Math. Soc.* (1) **35**, 3–27.

Macaulay, F. (1916): *The algebraic theory of modular systems.* Cambridge Univ. Press, Cambridge

Macaulay, F. (1927): Some properties of enumeration in the theory of modular systems. *Proc. London Math. Soc.* **26**, 531–555.

Matsumura, H. (1986): *Commutative ring theory.* Cambridge Univ. Press, Cambridge.

Mayr, E.; Meyer, A. (1982): The complexity of the word problem for commutative semigroups and polynomial ideals. *Adv. in Math.* **46**, 305–329.

McConnel, J.C.; Robson, J.C. (2001): *Noncommutative Noetherian Rings.* Graduate Studies in Mathematics, 30. AMS, Providence, RI.

Menabrea, L.F. (1842): Sketch of the analytical engine invented by Charles Babbage. With notes upon the memoir by the translator, Ada Augusta, Countess of Lovelace. In: P. Morrison and E. Morrison (eds.), *Charles Babbage and his calculating engines,* part II, Dover Publications, New York (1961).

Milnor, J. (1968): *Singular points of complex hypersurfaces.* Princeton Univ. Press.

Molien, T. (1897): Über die Invarianten der linearen Substitutionsgruppen. *Sitzungsber. König. Preuss. Akad. Wiss.*, 1152–1156.

Möller, H.M. (1993): On decomposing systems of polynomial equations with finitely many solutions. *Appl. Algebra Eng. Commun. Comput.* **4**, 217–230.

Möller, H.M. (1998): Gröbner Bases and Numerical Analysis. In: B. Buchberger and F. Winkler (eds.): *Gröbner Bases and Applications,* 159–178, LNS 251, CUP, Cambridge.

Möller, H.M.; Mora, F. (1984): Upper and lower bounds for the degree of Gröbner bases. In: *Proceedings EUROSAM 84 (Cambridge, 1984)*, Lecture Notes in Comput. Sci. 174, 172–183, Springer-Verlag.

Mora, T. (1982): An algorithm to compute the equations of tangent cones. In: *Computer algebra, Proceedings EUROCAM '82, Marseille,* 158–165, Springer-Verlag.

Mora, T. (1986): Gröbner bases for non-commutative polynomial rings. In: *Proc. AAECC 3*, Lect. N. Comp. Sci. **229**, 353–362.

Mora, T. (1994): An introduction to commutative and noncommutative Gröbner bases. *Theor. Comput. Sci.* **134**, No. 1, 131–173.

Mumford, D. (1970): *Abelian varieties.* Tata Institute of Fundamental Research Studies in Mathematics. 5. Oxford University Press. VIII.

Mumford, D.; Fogarty, J.; Kirwan, F. (1994): *Geometric invariant theory.* Third edition. Springer-Verlag.

Mumford, D. (1999): *The red book of varieties and schemes.* Second, expanded edition. Lecture Notes in Math. 1358, Springer-Verlag.

Newstead, P. (1978): *Introduction to Moduli Problems and Orbit Spaces.* Springer-Verlag.

Noether, E. (1921): Idealtheorie in Ringbereichen. *Math. Ann.* **83**, 24–66.
Noether, E. (1926): Der Endlichkeitssatz der Invarianten endlicher linearer Gruppen der Charakteristik *p. Nachr. v. d. Ges. d. Wiss. zu Göttingen*, 28–35.
Nolan, J.F. (1953): *Analytical differentiation on a digital computer.* Master Thesis, MIT, Cambridge.
Okonek, C. (1983): Moduli reflexiver Garben und Flächen von kleinem Grad in $\mathbb{P}^4$. *Math. Z.* **184**, 549–572.
Okonek, C.; Schneider, M.; Spindler, H. (1980): *Vector bundles on complex projective spaces.* Birkhäuser, Boston.
Pedersen, P.; Sturmfels, B. (1993): Product formulas for resultants and Chow forms. *Math. Z.* **214**, No. 3, 377–396.
Popescu, S. (1993): *On smooth surfaces of degree* ≥ 11 *in* $\mathbb{P}^4$. Dissertation, Universität des Saarlandes, Saarbrücken.
Popescu, S.; Ranestad, K. (1996): Surfaces of degree 10 in the projective fourspace via linear systems and linkage. *J. Alg. Geom.* **5**, 13–76.
Reid, M. (1988): *Undergraduate Algebraic Geometry.* Cambridge University Press, Cambridge.
Richman, D.R. (1990): On vector invariants of finite fields. *Adv. in Math.* **81**, 30–65.
Risch, R. (1968): On the integration of elementary functions which are built up using algebraic operations. *Report SP-2801/002/00*, System Development Corp., Santa Monica.
Risch, R. (1969a): Further results on elementary functions. *Report RC-2402*, IBM Corp., Yorktown Heights.
Risch, R. (1969b): The problem of integration in finite terms. *Trans. A.M.S.* **139**, 167–189.
Risch, R. (1970): The solution of the problem of integration in finite terms. *Bull. A.M.S.* **76**, 605–608.
Robbiano, L. (1985): Term Orderings on the Polynomial Ring. In: *Proceedings EUROCAL 85*, LNCS 204, Springer-Verlag.
Roelse, P. (1999): Factoring high-degree polynomials over $\mathbb{F}_2$ with Niederreiter's algorithm on the IBM SP2. *Math. Comp.* **68**, 869–880.
Rybowicz, M. (1990): *Sur le calcul des places et des anneaux d'entiers d'un corps de fonctions algébriques.* Thèse d'état, Univ. de Limoges.
Saito, K. (1971): *Quasihomogene isolierte Singularitäten von Hyperflächen.* Invent. Math. **14**, 123–142.
Schreyer, F.-O. (1980): *Die Berechnung von Syzygien mit dem verallgemeinerten Weierstraßchen Divisionssatz und eine Anwendung auf analytische Cohen–Macaulay Stellenalgebren minimaler Multiplizität.* Diplomarbeit, Universität Hamburg.
Schreyer, F.-O. (1991): A Standard Basis Approach to Syzygies of Canonical Curves. *J. Reine Angew. Math.* **421**, 83–123.
Schreyer, F.-O. (1996): Small fields in constructive algebraic geometry. In: M. Maruyama (ed.), *Moduli of vector bundles,* 221–228, Marcel Dekker.
Schreyer, F.-O.; Tonoli, F. (2002): Needles in a haystack: Special varieties via small fields. In: Eisenbud et al (2002), 251–279.
Shafarevich, I.R. (1974): *Basic Algebraic Geometry.* Springer-Verlag.
Shimoyama, T.; Yokoyama, K. (1996): Localization and primary decomposition of polynomial ideals. *J. Symb. Comput.* **22**, 247–277.

Silhol, R. (1978): Géometrie algébrique sur un corps non algébriquement clos. *Commun. Alg.* **6**, 1131–1155.

Slodowy, P. (1980): *Simple singularities and simple algebraic groups.* Lecture Notes in Math. 815, Springer-Verlag.

Smith, G.G. (2000): Computing global extension modules. *J. Symb. Comput.* **29**, 729–746.

Smith, K.E.; Kahanpää, L.; Kekäläinen, P.; Traves, W. (2000): *An invitation to algebraic geometry.* Springer-Verlag.

Sommese, A.J.; Wampler, C.W. (2005): *The numerical solution of systems of polynomials arising in engineering and science.* World Scientific.

Sturmfels, B. (1993): *Algorithms in Invariant Theory.* Springer-Verlag.

Sturmfels, B. (1993a): Sparse elimination theory. In: D. Eisenbud et al (eds.), *Computational algebraic geometry and commutative algebra. Proceedings of a conference held at Cortona, Italy, 1991.* Cambridge University Press. Symp. Math. **34**, 264–298.

Sturmfels, B. (1996): *Gröbner bases and convex polytopes.* University Lecture Series, 8. AMS, Providence, RI.

Sturmfels, B. (1997): Introduction to resultants. In: D. Cox, B. Sturmfels (eds.), *Applications of Computational Algebraic Geometry.* Proceedings of Symp. in Applied Math., **53**, American Mathematical Society, 25–39.

Sturmfels, B. (2002): *Solving Systems of Polynomial Equations.* CBMS Regional Conference Series in Mathematics, AMS, Providence, RI.

Sylvester, J.J. (1904–1912): *The Collected Mathematical Papers of James Joseph Sylvester.* Cambridge University Press, Cambridge.

Trager, B.M. (1976): Algebraic factoring and rational function integration. In: R.D. Jenks (ed.), *Proc. SYMSAC '76*, 196–208, ACM press.

Tran, Q.-N. (2000): A Fast Algorithm for Gröbner Basis Conversion and its Applications. *J. Symb. Comput.* **30**, 451–467.

Traverso, C. (1996): Hilbert functions and the Buchberger algorithm. *J. Symb. Comput.* **22**, No. 4, 355–376.

van der Waerden, B. (1931): *Moderne Algebra, Vol. II.* Springer-Verlag. English translation: *Modern Algebra, Vol. II.* F. Ungar Publ. Co. New York (1950).

Wang, D. (1989): Characteristic sets and zero structures of polynomial sets. *Preprint RISC-LINZ*, Linz, Austria.

Wang, D. (2001): *Elimination methods.* Springer-Verlag.

Wester, M. (ed.) (1999): *Computer algebra systems. A practical guide.* Wiley, Chichester.

Weyman, J.; Zelevinsky, A. (1994): *Determinantal formulas for multigraded resultants. J. Alg. Geom.* **3**, 569–597.

Zariski, O.; Samuel, P. (1975–1976). *Commutative Algebra.* Vols. I and II. Corr. 2nd printing of the 1958–1960 edition. Springer-Verlag.

Zassenhaus, H. (1969): On Hensel factorization. *J. Number Theory* **1**, 291–311.

Index