Algorithms and Computation in Mathematics • Volume 16

Editors

Arjeh M. Cohen Henri Cohen
David Eisenbud Bernd Sturmfels

Wolfram Decker
Christoph Lossen

Computing in Algebraic Geometry

A Quick Start using SINGULAR

Springer

HINDUSTAN BOOK AGENCY

Wolfram Decker

Fachrichtung 6.1 Mathematik
Universität des Saarlandes
Postfach 151 150
66041 Saarbrücken, Germany
e-mail: decker@math.uni-sb.de

Christoph Lossen

Fachbereich Mathematik
Technische Universität Kaiserslautern
Postfach 30 49
67653 Kaiserslautern, Germany
e-mail: lossen@mathematik.uni-kl.de

Library of Congress Control Number: 2005938498

Mathematics Subject Classification (2000): 13P05, 13P10, 14Q99, 14A05, 14A25, 68W30, 11Y35, 12Y05, 13D02, 13D07, 13H10, 14B25, 16E05, 13A50, 14L24

ISSN 1431-1550

ISBN-10 3-540-28992-5 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-28992-0 Springer Berlin Heidelberg New York

Typesetting: by the authors using a Springer LaTeX macro package
Production: LE-TeX Jelonek, Schmidt & Vöckler GbR, Leipzig
Cover design: *design & production* GmbH, Heidelberg

Printed on acid-free paper 46/3100YL - 5 4 3 2 1 0

To Helmi, Doris, Anne, and Matthias, with love
W.D.

To Carmen, Katrin and Carolin, with love
C.L.

Preface

Systems of polynomial equations are central to mathematics and its application to science and engineering. Their solution sets, called algebraic sets, are studied in algebraic geometry, a mathematical discipline of its own. Algebraic geometry has a rich history, being shaped by different schools. We quote from Hartshorne's introductory textbook (1977):

> "Algebraic geometry has developed in waves, each with its own language and point of view. The late nineteenth century saw the function-theoretic approach of Brill and Noether, and the purely algebraic approach of Kronecker, Dedekind, and Weber. The Italian school followed with Castelnuovo, Enriques, and Severi, culminating in the classification of algebraic surfaces. Then came the twentieth-century "American school" of Chow, Weil, and Zariski, which gave firm algebraic foundations to the Italian intuition. Most recently, Serre and Grothendieck initiated the French school, which has rewritten the foundations of algebraic geometry in terms of schemes and cohomology, and which has an impressive record of solving old problems with new techniques. Each of these schools has introduced new concepts and methods."

As a result of this historical process, modern algebraic geometry provides a multitude of theoretical and highly abstract techniques for the qualitative and quantitative study of algebraic sets, without actually studying their defining equations at the first place.

On the other hand, due to the development of powerful computers and effective computer algebra algorithms at the end of the twentieth century, it is nowadays possible to study explicit examples via their equations in many cases of interest. In this way, algebraic geometry becomes accessible to experiments. The experimental method, which has proven to be highly successful in number theory, now also adds to the toolbox of the algebraic geometer.

As in other areas of pure mathematics, computer algebra may help

- to discover unexpected mathematical evidence, leading to new conjectures or theorems, later proven by traditional means,
- to construct interesting objects and determine their structure (in particular, to find counterexamples to conjectures),
- to verify negative results such as the nonexistence of certain objects with prescribed invariants,
- to verify theorems whose proof is reduced to straightforward but tedious calculations,
- to solve enumerative problems, and
- to create data bases.

There is a growing number of research papers in algebraic geometry originating from explicit computations. The computational methods also play a significant role when it comes to applications of algebraic geometry to practical problems. And, they enter the classroom, allowing us to introduce students at an early stage to algebraic geometry, without developing too much of its abstract machinery.

What are these Notes About ?

These notes are intended to provide a quick start to computing in algebraic geometry. For each topic treated, we include a compact presentation of the background material from commutative algebra and algebraic geometry needed to understand that topic. Further, we discuss the relevant algorithms and explain how to use them in studying algebraic sets. And, we present many explicit computational examples which simultaneously introduce the computer algebra system `SINGULAR` and which may serve as samples for computations carried out by the reader. By revealing implementation details, we point out how to access alternative algorithms for specific tasks. When applying the algorithms to concrete research problems, the difference in their performance could mean to get a result, or to run out of time or memory.

In our presentation, we essentially omit proofs, giving references to standard textbooks instead. Also, at the end of each chapter, we give hints on further reading. These refer to basic definitions and proofs, and to more advanced material as well.

Our main reason for focusing on a single computer algebra system is that we want to keep the size of the notes within reasonable limits. `SINGULAR`, the system of our choice for this purpose, offers a large variety of tools for computations in commutative algebra, algebraic geometry, and singularity theory. We use `SINGULAR 3-0` whose many new features have not yet been described in other textbooks. In fact, some of these features have been implemented by the authors together with other members of the `SINGULAR` team to make the examples presented in these notes work.

The notes originate from an intense one week course given by the authors at Allahabad, India, January 5–11, 2003 and from other schools taught by the authors. The Allahabad course started with an introductory lecture on computer algebra, and it ended with an additional lecture on computing sheaf cohomology and Beilinson monads, given on demand of some of the experienced members of the audience. In between, the authors gave a series of lectures in the morning, and posted practical exercises for the afternoon. The junior and senior participants worked on the exercises in front of the computers, with advice being given by the authors. It is worth pointing out that these practical sessions often ended well after midnight.

Who may Benefit from the Text

The text may accompany students taking a beginner's course in algebraic geometry who might wish to further explore their new playground by experimenting with examples according to their growing knowledge. It can also help Master and PhD students, as well as more experienced researchers, add powerful computational methods to their personal toolbox. Further, it may appeal to users of systems other than `SINGULAR` who might occasionally need techniques not implemented in their system of choice. In addition, we address students and researchers interested in implementing their own computational tools using `SINGULAR` as their basis. For these readers we explain, in particular, how to write and debug `SINGULAR` libraries. We believe that the use of this book is not restricted to students and researchers specializing in algebraic geometry itself, but will also prove useful to those in related disciplines.

The Structure of the Text

Although this text widely extends the written material presented at Allahabad, we maintained the original structure of the course, organizing the material as an introductory lecture, Lectures 1–9, Practical Sessions I–V, and an appendix.

We start in the introductory lecture by giving some remarks on the development of computer algebra. On our way, we present several computer algebra sessions featuring some of the systems which are relevant for researchers in algebraic geometry and related fields.

Lecture 1 is an introduction by historical remarks to the concept of Gröbner bases which is fundamental to computational algebraic geometry. In Lecture 2, we discuss basic computational problems arising from the geometry-algebra dictionary and their solution by means of Gröbner basis methods. Lectures 1 and 2 both already contain explicit `SINGULAR` examples which may serve as samples for those wishing to make their first computational experiments with `SINGULAR`. A thorough introduction to `SINGULAR` is given in Lecture 3. This introduction widely exceeds what is necessary for taking the first steps into `SINGULAR` as it should also serve as a reference for more experienced users. In the Allahabad course, Lecture 3 was divided into two parts, with Practical Session I being held right after the first part, and Practical Session II right after the second part.

Lectures 4 and 5 treat computations in homological algebra, covering basic constructions such as kernels, cokernels, Hom, Ext and Tor, and the more advanced concepts of flatness and Cohen-Macaulay rings. Such computations take center stage in Practical Session III.

In Lecture 6, we discuss some of the methods for exact and symbolic-numerical solving of systems of polynomial equations. These include decomposition techniques for algebraic sets. Primary decomposition is treated separately in Lecture 7 which also deals with normalization. Next follows Practical Session IV.

In Lecture 8, we return to the historical origin of Gröbner bases as presented in Lecture 1, giving an overview on recent algorithms for invariant theory.

Lecture 9 is dedicated to the local study of algebraic sets and, thus, to computations in local rings. For this, we extend the concept of Gröbner bases by introducing standard bases. Corresponding exercises can be found in Practical Session V.

In the appendix, we include the additional lecture on computing sheaf cohomology and Beilinson monads, and we give solutions to the exercises posted in Practical Sessions I–V.

The Level of the Text

Since we address students and researchers, the level of the text is necessarily uneven. Some familiarity with groups, rings, ideals, fields, and vector spaces, together with the information provided in these notes, will enable the reader to understand many of the computations in Lectures 1, 2, 3, 6, 7, and 8. Though we summarize some of the basic concepts of algebraic geometry to provide a common language for all readers, some familiarity with these concepts is needed to fully appreciate our geometric interpretation of the computations. The absolute beginner should, thus, read these notes in conjunction with other books such as Reid's "Undergraduate Algebraic Geometry" (1988) or the undergraduate text "Ideals, Varieties, and Algorithms" by Cox, Little, and O'Shea (1997).

Even in the more elementary lectures, the unexperienced reader will find mathematical statements for which he is not prepared. In contrast, the computational recipes arising from the statements are often easy to understand. The reader who is willing to take the recipes for granted will have no problems in applying them to study algebraic sets.

For large parts of Lectures 4, 5, and 9, for the second section of Lecture 7, for one example in Lecture 8, and for Appendix A, more background in commutative algebra and algebraic geometry is needed.

Exercises

The exercises are designed so as to make the beginner familiar with some basic features of computational algebraic geometry and `SINGULAR`. The serious reader should solve each exercise in front of the computer before turning to the authors' solution of that exercise in the appendix. Further, we highly recommend to check the textbooks by Cox, Little, and O' Shea (1997, 1998), Greuel and Pfister (2002), and Decker and Schreyer (2006) for further exercises admitting a `SINGULAR` solution.

Basic Conventions

If not otherwise mentioned, each **ring** considered in these notes is commutative, and it has a multiplicative identity 1. Ring homomorphisms take 1 to 1. If R is a subring of a ring S, and if I is an ideal of R, we write $I \cdot S$ or simply IS for the ideal generated by I in S. In the context of free resolutions, we often write $N \longleftarrow M$ for a homomorphism $M \longrightarrow N$ since this fits well with how `SINGULAR` displays numerical information on free resolutions.

We work over a field K, referring to the elements of K as **scalars**. We usually write $K[\boldsymbol{x}] = K[x_1, \ldots, x_n]$ for the polynomial ring in n variables over K. A **monomial** in $K[\boldsymbol{x}]$ is a product $\boldsymbol{x}^\alpha = x_1^{\alpha_1} \cdots x_n^{\alpha_n}$, where $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n$. A **term** in $K[\boldsymbol{x}]$ is a scalar times a monomial.

Timings and `SINGULAR` Output

Occasionally, we print the CPU time used by a `SINGULAR` computation. All timings are given in full seconds, taken on a Pentium IV 2.4 GHZ processor.

In documenting `SINGULAR` sessions, we print `[...]` to indicate that part of the `SINGULAR` output is omitted. Without printing `[...]`, we omit the output displayed by `SINGULAR` when loading a library. This output gives information on the library loaded and on related libraries which are automatically loaded as well.

Website

We maintain a website for this book at

`http://www.singular.uni-kl.de/BOOK_DL/`

We are grateful for comments and corrections which will be posted at the website if appropriate. Also, selected pieces of code written for the book can be downloaded from the website.

Acknowledgment

The Allahabad school was a joint activity of the Harish-Chandra Research Institute at Allahabad and Bhaskaracharya Pratishthana at Pune. We greatly appreciated the facilities provided by the Harish-Chandra Research Institute, and we are grateful to C. S. Inamdar, the Custodian of Bhaskaracharya Pratishthana. The school was partially supported by the National Board for Higher Mathematics, India. We would like to thank all participants of the Allahabad school and the other schools we taught over the years for their enthusiasm and the fun we had together. Special thanks go to S.D. Adhikari, S.A. Katre, R.A. Rao, D.N. Sheth and J.K. Verma for inviting us to India and for plenty of feedback and help. We are particularly grateful to M. Joshi for guiding us to some of the Allahabad sights, including the confluence of the two rivers, and for patiently answering so many questions on Indian culture. Further, we very much enjoyed a wonderful concert with classical Indian music, featuring, among others, our colleagues M.V. Manohar and A.R. Shastri.

We would like to thank G. Pfister, H. Schönemann, and F.-O. Schreyer for many helpful discussions, G. Lecerf for sharing his ideas on polynomial factorization and for writing the library `absfact.lib`, W. Bruns for pointing out several references to us, and A. Frühbis-Krüger, M. Kunte, V. Levandovskyy, T. Markwig, T. von Oertzen, and J. Webber for reading parts of the manuscript and making valuable suggestions. Last but not least, the first author is grateful to the MSRI at Berkeley for its hospitality while working on the final version of the book.

Saarbrücken, Kaiserslautern,
October 2005

Wolfram Decker
Christoph Lossen

Contents

Introductory Remarks on Computer Algebra

The remarks in this lecture address computer algebra, its history, and computer algebra systems. We give a few examples of what can be computed, focusing on some of the systems which are relevant for researchers in algebraic geometry and related fields. Typically, the examples aim at the experienced reader. They will not play a role in the subsequent lectures.

Most of mathematics is concerned at some level with setting up and solving equations, for example to model applications in science and engineering. In many cases, this involves tedious computations which are difficult to get right or too extensive to be carried through by hand. Two mathematical disciplines of their own, numerical analysis and computer algebra, originate from this problem. In contrast to numerical analysis, calculations in computer algebra are carried through exactly, that is, no approximation is applied at any step. Infinite precision arithmetic allows one to compute in the ring of integers, in the field of rationals and in algebraic number fields, in finite prime fields and in arbitrary Galois fields, in polynomial rings and in fields of rational functions, in difference fields (needed for indefinite summation), and in differential fields (needed for indefinite integration). In fact, there is a much larger variety of algebraic structures in which algebraic algorithms allow one to manipulate algebraic objects or the structures themselves. Exact computer algebra methods enable us to create algorithms that decide, for example, the solvability of systems of polynomial equations or the solvability of indefinite summation and integration problems in certain specified classes of functions (see von zur Gathen and Gerhard (1999) and the references cited there).

That automated computing is not restricted to numerical computation was already evident to Charles Babbage and Lady Ada Augusta in the 19th century (see Larcombe (1999) for the story). Besides working on his Difference Engines, Babbage spent many years of his life on designing a more universal mechanical machine (Analytical Engine), to be programmed with punch-cards invented by Josef-Maria Jacquard for the control of automatic looms. Babbage's earliest ideas on how such a machine could perform automated algebraic manipulations are documented in his notebook (1836):

> “This day I had for the first time a general but very indistinct conception of the possibility of making an engine work out *algebraic* developments – I mean without *any* reference to the *value* of the letters. My notion is that as the cards (Ja[c]quards) of the calc. engine direct a series of operations and then recommence with the first, so it might perhaps be possible to cause the same cards to punch others equivalent to any given number of repetitions. But these hole[s] might perhaps be small pieces of formulae previously made by the first cards ...”

And, Lady Ada wrote (see Menabrea (1842)):

> “There are many ways in which it may be desired in special cases to distribute and keep separate the numerical values of different parts of an algebraical formula; and the power of effecting such distributions to any extent is essential to the *algebraical* character of the Analytical Engine. Many persons who are not conversant with mathematical studies, imagine that because the business of the engine is to give its results in *numerical notation*, the *nature of its processes* must consequently be *arithmetical* and *numerical*, rather than *algebraical* and *analytical*. This is an error. The engine can arrange and combine its numerical quantities exactly as if they were *letters* or any other *general* symbols; and in fact it might bring out its results in algebraical *notation*, were provisions made accordingly. It might develop three sets of results simultaneously, viz. *symbolic* results ...; *numerical* results (its chief and primary object); and *algebraical* results in *literal* notation.”

The Analytical Engine, however, was never built, and symbolic computations were not carried through in an automated way until electronic computers were available. The first documented computer algebra programs, written for analytic differentiation, were described in two Master Theses by Kahrimanian (1953) and Nolan (1953). According to the historical account in the introductory textbook by Geddes, Czapor, and Labahn (1992), the beginning 1960’s saw increasing activities in the field, in particular, after the LISP language,

> “a major advancement on the road to languages for symbolic computation”,

had been developed. In the period between 1961 and 1971,

> “the field progressed from birth through adolescence to at least some level of maturity.”

This progress may be marked by some algorithmic breakthroughs, and by the first releases of some well-known LISP-based general purpose computer algebra systems:

1965	Buchberger's algorithm for computing Gröbner bases.
1967, 1970	Berlekamp's algorithm for factorizing univariate polynomials over finite fields.
1968-1970	Risch's algorithm for the indefinite integration of elementary functions.
1969	Zassenhaus' algorithm for factorizing univariate polynomials over the integers.
1968, 1970	First releases of REDUCE, respectively REDUCE2.
1970	First release of MACSYMA.
1971	First release of SCRATCHPAD (today known as AXIOM).

Nowadays, there is a large variety of computer algebra systems suiting different needs. User-friendly interfaces and comfortable help functions enable us to work with powerful computing tools by just looking up a few commands, without any knowledge in programming. On the other hand, modern systems also offer a user language which allows those interested in programming to extend the system. Many user-written packages and libraries are publicly available, providing, thus, even more computing tools.

There are general purpose and special purpose computer algebra systems. Typically, one has to pay for a general purpose system whereas many of the special purpose systems can be downloaded from the internet for free. A general purpose system allows one to attack problems in many different areas. In addition to tools for symbolic computation, such a system usually offers tools for numeric computation and for visualization. Well-established general purpose systems are REDUCE, MACSYMA, MAPLE, DERIVE, MATHEMATICA, MUPAD, and AXIOM (see Wester (1999) for a critical comparison).

Example 1. MAPLE, firstly released in 1983, is a general purpose system with a kernel written in C, and with most higher level functions or packages written in the MAPLE user language. The following MAPLE session features polynomial factorization, indefinite summation, and indefinite integration:

```
> factor(y^2-x*y-x^2*y+x^3);
                                            2
                           (-y + x) (-y + x )

> sum((-1)^k*binomial(n,k), k=0..m);

                              (m + 1)
                  (m + 1) (-1)        binomial(n, m + 1)
                - --------------------------------------
                                    n

> int( x/(x^2-1), x );

                    1/2 ln(x - 1) + 1/2 ln(x + 1)
```
□

Today, specialized algorithms and software allow one to factorize randomly chosen polynomials in $\mathbb{F}_2[x]$ of degree 300 000 (see Roelse (1999)).

There are plenty of user-written MAPLE packages[1] some of which are designed for applications in algebraic geometry.

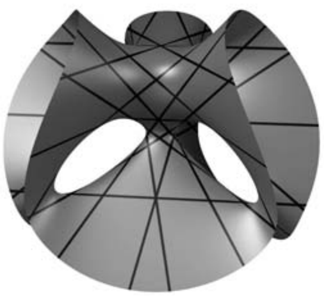
Example 2. SCHUBERT is a MAPLE package specializing on computations in intersection theory. Having downloaded and installed SCHUBERT, one can load it into a MAPLE session:

```
with(SF): # symmetric functions package, obtained from Schubert's webpage
with(schubert);
[\&*, \&-!, \&-*, \&/, \&^*, End, Grass, Hom, POINT, Proj, Symm, adams,
  additivebasis, betti, blowup, blowuppoints, bundle, bundlesection,
  chern, chi, codimension, compose, curve, determinant, dimension,
  division, down, dual, grass, grobnerbasis, insertedge, integral,
  integral2, koszul, lowershriek, lowerstar, monomials, monomialvalues,
  morphism, multiplepoint, normalbundle, normalform, o, porteous,
  porteous2, productvariety, proj, rank, schur, schurfunctor,
  schurfunctor2, segre, setvariety, sheaf, strip, symm, tangentbundle,
  tensor, todd, toddclass, toricvariety, totalspace, twist, up, upperstar,
  variety, verifyduality, wedge, where, whichcone, wproj]
```

Using SCHUBERT, we compute the number of lines on a general cubic surface in projective 3-space $\mathbb{P}^3$:

```
grass(2,4,c);            # Lines in P^3.

                        currentvariety_ is Gc, DIM is 4

B := symm(3,Qc):         # Qc is the rank 2 quotient bundle, B its
                         # 3rd symmetric power.
c4 := chern(rank(B),B):  # the 4th Chern class of this rank 4 bundle.
integral(c4);
                                    27
```

Clebsch's diagonal cubic is a particular nice example of a cubic surface in $\mathbb{P}^3$. We use the software SURF to visualize the diagonal cubic and the lines on it:



□

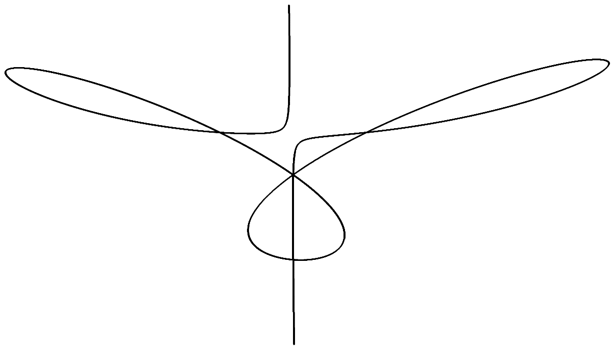
[1] The use of such a package may require an older version of MAPLE.

Example 3. Another `MAPLE` package which can be downloaded for free is `CASA`. It offers various tools for algebraic geometers.

```
> with(casa);
           |__|
           |  |          Welcome to CASA 2.5 for Maple V.5
     |   /\|  |/\
    /=\__|  []  |
   /            \_       Copyright (C) 1990-2000 by Research Institute
   |              \      for Symbolic Computation (RISC-Linz), the
   \   CASA 2.5   |      University of Linz, A-4040 Linz, Austria.
    |             |
   _|     |||     |      For help type '?casa' or '?casa,<topic>'.
__/       |||     |_

[BCH2, BCHDecode, CyclicEncode, DivBasisL, GWalk, GoppaDecode, GoppaEncode,
   GoppaPrepareDu, GoppaPrepareSV, GoppaPrepareSa, GoppaPrimary,
   Groebnerbasis, InPolynomial, NormalPolynomial, OutPolynomial,
   PolynomialRoots, RPHcurve, SakataDecode, SubsPolynomial, _casaAlgebraicSet,
   adjointCurve, algset, casaAttributes, casaVariable, computeRadical, conic,
   decompose, delete, dimension, equalBaseSpaces, equalProjectivePoints,
   finiteCurve, finiteField, generators, genus, homogeneousForm,
   homogeneousPolynomial, homogenize, implDifference, implEmpty, implEqual,
   implIdealQuo, implIntersect, implOffset, implSubSet, implUnion,
   implUnionLCM, imult, independentVariables, init, isProjective, leadingForm,
   makeDivisor, mapOutPolynomial, mapSubsPolynomial, mgbasis, mgbasisx,
   mkAlgSet, mkImplAlgSet, mkParaAlgSet, mkPlacAlgSet, mkProjAlgSet, mnormalf,
   msolveGB, msolveSP, mvresultant, neighbGraph, neighborhoodTree,
   numberOfTerms, pacPlot, paraOffset, parameterList, passGenCurve,
   planecurve, plotAlgSet, pointInAlgSet, projPoint, properParametrization,
   properties, rationalPoint, realroot_a, realroot_sb, setPuiseuxExpansion,
   setRandomParameters, singLocus, singularities, ssiPlot, subresultantChain,
   tangSpace, toAffine, toImpl, toPara, toPlac, toProj, toProjective, tsolve,
   variableDifferentFrom, variableList]
```

We use `CASA` to visualize and parametrize a rational plane curve given by its equation:

```
> f := 1/4*y^3+x^2*y^3-2*x^3*y^2-1/4*x^4*y+1/4*x^4-5/4*x^2*y^2+43/4*x*y^3
>      +1/4*y^5+1/2*y^4+1/2*x^5+20*x*y^4+37/8*x^3-39/4*x^3*y-37/4*x*y^2;
          3    2  3      3  2        4          4        2  2           3
f := 1/4 y  + x  y  - 2 x  y  - 1/4 x  y + 1/4 x  - 5/4 x  y  + 43/4 x y

          5        4        5         4         3         3             2
   + 1/4 y  + 1/2 y  + 1/2 x  + 20 x y  + 37/8 x  - 39/4 x  y - 37/4 x y

> A := mkImplAlgSet([f],[x,y]):
> Ap := plotAlgSet(A,x=-4..4,y=-2..2,numpoints=10000,thickness=2,axes=none,
>                    colour=black,scaling=constrained):
> with(plots):
> plotsetup(ps,plotoutput=`curve.ps`,plotoptions=`portrait,noborder`);
```



```
> toPara(A,t);
Parametric_Algebraic_Set([

        4          2          3        5
  1443 t  - 2888 t  - 2 - 37 t  + 37 t  - 152 t
2 ---------------------------------------------,
            2       4       3                 5
  16 + 180 t  - 16 t  - 12 t  + 644 t + 1365 t

            4          3         2                 5
   -8 + 74 t  - 2590 t  - 222 t  - 300 t + 1369 t
 - ----------------------------------------------], [t])
              2       4       3                 5
    16 + 180 t  - 16 t  - 12 t  + 644 t + 1365 t
```

The example is due to J. Böhm. We refer to his Diploma Thesis (1999) for further examples and for a nice survey on algorithms for parametrizing rational curves. □

For some of the more special and advanced applications, general purpose systems are not powerful enough. Often, the implementation of the required algorithms is not optimal with respect to speed and storage handling; in addition, some of the more advanced algorithms might not be implemented at all. Many of today's special purpose systems have been created by people specializing in a field other than computer algebra and having a desperate need for computational power in the context of some of their research problems. A pioneering and prominent example is Veltman's SCHOONSCHIP which helped to win a Nobel price in physics in 1999 (awarded to Veltman and t'Hooft "for having placed particle physics theory on a firmer mathematical foundation"). From a special purpose system, we expect highly tuned implementations of the algorithms needed for the area in which the system is specializing. Examples are KANT, LIDIA, MAGMA (which is not free), PARI and SIMATH for number theory, or GAP and MAGMA for group theory.

Example 4. In the following LIDIA session, we factor the 8th Fermat number $F_8 = 2^{2^8} + 1$:

```
lc> factor(2^(2^8)+1);
$0 =  [(1238926361552897,1)
(93461639715357977769163558199606896584051237541638188580280321,1)]
```

□

At this writing, the 11th Fermat number is the largest Fermat number all whose factors are known.

Example 5. We use GAP to compute the character table of the Heisenberg group H_3. In its Schrödinger representation, H_3 is the subgroup of $\mathrm{GL}_3(\mathbb{C})$ generated by the matrices

$$\sigma = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \quad \text{and} \quad \tau = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \xi & 0 \\ 0 & 0 & \xi^2 \end{pmatrix}.$$

Here, $\xi = e^{\frac{2\pi i}{3}}$ is a primitive 3rd root of unity in $\mathbb{C}$.

```
gap> m1 := [[0,0,1],[1,0,0],[0,1,0]];;
gap> m2 := [[1,0,0],[0,E(3),0],[0,0,E(3)^2]];;
gap> G := Group(m1,m2);;
gap> Size(G);
27
gap> Display(CharacterTable(G));
CT1
      3  3  2  2  2  2  2  2  2  2  3  3

         1a 3a 3b 3c 3d 3e 3f 3g 3h 3i 3j

X.1       1  1  1  1  1  1  1  1  1  1  1
X.2       1  A  A  A /A /A /A  1  1  1  1
X.3       1 /A /A /A  A  A  A  1  1  1  1
X.4       1  1 /A  A  1 /A  A /A  A  1  1
X.5       1  A  1 /A /A  A  1 /A  A  1  1
X.6       1 /A  A  1  A  1 /A /A  A  1  1
X.7       1  1  A /A  1  A /A  A /A  1  1
X.8       1  A /A  1 /A  1  A  A /A  1  1
X.9       1 /A  1  A  A /A  1  A /A  1  1
X.10      3  .  .  .  .  .  .  .  .  B /B
X.11      3  .  .  .  .  .  .  .  . /B  B

A = E(3)
  = (-1+ER(-3))/2 = b3
B = 3*E(3)^2
  = (-3-3*ER(-3))/2 = -3-3b3
```

Via the online GAP user manual, one easily finds out how to read the output. □

Using GAP, finite groups could be, essentially, classified up to order 2000 (see Besche et al (2001)). For instance, there are 10 494 213 groups of order 512 (up to isomorphism).

Example 6. We use MAGMA to compute a fundamental system of invariants of the Heisenberg group H_3 in its Schrödinger representation:

```
> R<x> := PolynomialRing(Integers());
> K<g> := NumberField(x^2+x+1);
> m1 := [0,0,1,1,0,0,0,1,0];
> m2 := [1,0,0,0,g,0,0,0,g^2];
> G := MatrixGroup<3,K|m1,m2>;
> R := InvariantRing(G);
> FundamentalInvariants(R);
[
    x1^3 + x2^3 + x3^3,
    x1*x2*x3,
    x1^6 + x2^6 + x3^6,
    x1^6*x3^3 + x1^3*x2^6 + x2^3*x3^6
]
```
□

Special purpose systems for commutative algebra and algebraic geometry allow us to manipulate ideals in polynomial rings (and much more). They typically rely on Gröbner basis techniques, and their engine is Buchberger's algorithm for computing Gröbner bases and syzygies. The pioneering MACAULAY and the more modern and complete COCOA, MACAULAY2, RISA/ASIR, and SINGULAR offer a variety of tools for experiments. FGB is a system just for the basic task of computing Gröbner bases. With BERGMAN, one can compute Gröbner bases, Hilbert series and Poincaré series in commutative and non-commutative graded algebras.

Example 7. We compute a Gröbner basis using SINGULAR:

```
                    SINGULAR                             /
 A Computer Algebra System for Polynomial Computations  /   version 3-0-1
                                                      0<
     by: G.-M. Greuel, G. Pfister, H. Schoenemann       \   October 2005
FB Mathematik der Universitaet, D-67653 Kaiserslautern   \

> ring R = 0, (x,y), dp;
> ideal I = xy, x2+y2;
> groebner(I);
_[1]=xy
_[2]=x2+y2
_[3]=y3
```

In MACAULAY2, the same Gröbner basis is obtained as follows:

```
Macaulay 2, version 0.9.2
--Copyright 1993-2001, D. R. Grayson and M. E. Stillman
--Singular-Factory 1.3b, copyright 1993-2001, G.-M. Greuel, et al.
--Singular-Libfac 0.3.2, copyright 1996-2001, M. Messollen

i1 : R = QQ[x,y]
o1 = R
o1 : PolynomialRing
i2 : I = ideal(x*y, x^2+y^2)
                    2    2
o2 = ideal (x*y, x  + y )
o2 : Ideal of R
i3 : gb I
o3 = | xy x2+y2 y3 |
o3 : GroebnerBasis
```

□

Remark 8 (Webpages). With the exception of SURF, the webpages of the systems and packages mentioned in this lecture can be easily found on the net. For SURF, see `http://sourceforge.net/projects/surf` .

Lecture 1

Basic Notations and Ideas: A Historical Account

The geometry-algebra dictionary, which will be studied in Lecture 2, translates geometric problems into algebraic problems and vice versa. It makes, in particular, many basic operations of algebraic geometry accessible to computer algebra methods. The work horse behind these methods is Buchberger's algorithm for computing Gröbner bases. In this lecture, we introduce Gröbner bases. The roots of Gröbner bases can be traced back to 19th century papers in classical invariant theory and, thus, to the roots of algebraic geometry itself. Proceeding by historical remarks, we include a discussion of several major ideas of Hilbert which are fundamental to commutative algebra and algebraic geometry and which are needed in subsequent lectures.

Classical invariant theory originated from the interest in properties of geometric objects which are unaffected by a change of coordinates. The question of how to represent such a property lead to the study of polynomials which are invariant under certain classes of coordinate transformations (see Lecture 8 for the basic setting of invariant theory). The property of being invariant under transformations of a given class is preserved under the ring operations on polynomials. The set of all invariants in a given situation inherits, thus, a ring structure from the polynomial ring. In fact, the invariants form a graded algebra over the given coefficient field.

Let K be a field. Each element of the polynomial ring $K[\boldsymbol{x}] = K[x_1, \dots, x_n]$ has a unique representation as a sum of homogeneous polynomials of different degrees. This is to say, the polynomial ring can be written as a direct sum

$$K[\boldsymbol{x}] = \bigoplus_{\nu \geq 0} K[\boldsymbol{x}]_\nu \,,$$

where $K[\boldsymbol{x}]_\nu$ is the K-vector space formed by the homogeneous polynomials of degree ν. The notion of a graded K-algebra is obtained by abstracting this:

Remark-Definition 1.1. A **graded ring** is a ring R together with a decomposition $R = \bigoplus_{\nu \geq 0} R_\nu$ as Abelian groups such that $R_\nu R_\mu \subset R_{\nu+\mu}$ for all ν, μ. A **homogeneous** element of R is an element f of some graded piece R_ν, and ν is then called the **degree** of f. An **ideal** of R is **homogeneous** if it is generated by homogeneous elements. If $R_0 = K$ is a field, then R is a K-algebra and the graded pieces R_ν are K-vector spaces. In this case, we say that R is a **graded $\boldsymbol{K}$-algebra**. □

We usually assume that R is a **finitely generated** (graded) **$\boldsymbol{K}$-algebra**. That is, there are (homogeneous) $f_1, \ldots, f_n \in R$ such that each element of R is a polynomial expression in the f_i. In this case, the R_ν are *finite dimensional* K-vector spaces.

Example 1.2. If I is a homogeneous ideal of $K[\boldsymbol{x}]$, the quotient ring $K[\boldsymbol{x}]/I$ inherits a graded structure from $K[\boldsymbol{x}]$. It is, thus, a (finitely generated) graded K-algebra. □

Besides explicitly computing particular invariants, the founders of classical invariant theory were interested in techniques to enumerate invariants. Cayley and Sylvester, for instance, made use of suitable representations of what is nowadays called the Hilbert series to give formulas for the dimension of the graded pieces of certain rings of invariants (see Cayley (1889) and J.J. Sylvester (1864)).

We define the Hilbert series in the more general setting of modules which are to rings what vector spaces are to fields. That is, a **module** over a ring R is an additively written Abelian group M, together with an operation $R \times M \to M$ such that for all $r, s \in R$ and $m, n \in M$ the following hold:

$$r(sm) = (rs)m,\ r(m+n) = rm + rn,\ (r+s)m = rm + sm,\ 1m = m\,.$$

Ideals I of $K[\boldsymbol{x}]$ and quotient rings $K[\boldsymbol{x}]/I$ are basic examples of $K[\boldsymbol{x}]$-modules which are studied in algebraic geometry. By speaking of modules we may often formulate definitions and results such that they apply to I and $K[\boldsymbol{x}]/I$ at the same time. Further, they apply to other modules which appear naturally in algebraic geometry.

Notions such as homomorphisms of R-modules, direct sums of R-modules, and submodules of R-modules are defined in the obvious way. If $m_1, \ldots, m_r$ are elements of a given R-module M, the set $\langle m_1, \ldots, m_r \rangle$ of all R-linear combinations of the m_i is a submodule of M to which we refer as the **submodule generated by the $\boldsymbol{m_i}$**. We usually assume that M is a **finitely generated $\boldsymbol{R}$-module**. That is, there are $m_1, \ldots, m_r \in R$ such that $M = \langle m_1, \ldots, m_r \rangle$.

Definition 1.3. *Let $R = \bigoplus_{\nu \geq 0} R_\nu$ be a graded ring. A* **graded module** *over R is an R-module M together with a decomposition $M = \bigoplus_{\nu \in \mathbb{Z}} M_\nu$ as Abelian groups such that $R_\nu M_\mu \subset M_{\nu+\mu}$ for all ν, μ. The elements of a graded piece M_ν are called* **homogeneous** *(of* **degree** *ν), and a* **submodule** *of M is called* **graded** *if it is generated by homogeneous elements.* □

If R is a finitely generated graded K-algebra and M is a finitely generated graded R-module, the graded pieces M_ν are finite dimensional K-vector spaces.

Definition 1.4. *Let R be a finitely generated graded K-algebra, and let $M = \bigoplus_{\nu \in \mathbb{Z}} M_\nu$ be a finitely generated graded R-module. The function*

$$H(M, \_\_) : \mathbb{Z} \longrightarrow \mathbb{Z}, \quad \nu \longmapsto \dim_K M_\nu \, ,$$

is called the **Hilbert function** *of M. The formal Laurent series*

$$H_M(t) := \sum_{\nu \in \mathbb{Z}} H(M, \nu) \cdot t^\nu \in \mathbb{Z}[[t, t^{-1}]]$$

is called the **Hilbert series** *of M.* □

Example 1.5. The monomials of degree ν in $K[\boldsymbol{x}] = K[x_1, \dots, x_n]$ form a K-vector space basis for $K[\boldsymbol{x}]_\nu$. Counting these monomials, we see that

$$H(K[\boldsymbol{x}], \nu) = \binom{\nu + n - 1}{n - 1}$$

(this formula holds for all $\nu \in \mathbb{Z}$ if we set $K[\boldsymbol{x}]_\nu = 0$ for $\nu < 0$). Thus, $H(K[\boldsymbol{x}], \nu)$ agrees for $\nu \geq -(n-1)$ with the polynomial expression

$$\frac{(\nu + n - 1)(\nu + n - 2) \cdots (\nu + 1)}{(n-1)!} \, .$$

We refer to this fact by saying that $H(K[\boldsymbol{x}], \_\_)$ is of **polynomial nature**. It means, in particular, that we may represent the infinitely many values of the Hilbert function of $K[\boldsymbol{x}]$ in finite terms. An alternative way of doing this is to write the Hilbert series as a rational function:

$$H_{K[x_1, \dots, x_n]}(t) = \sum_{\nu \in \mathbb{Z}} \binom{\nu + n - 1}{n - 1} \cdot t^\nu = \frac{1}{(1-t)^n} \, .$$

From this representation, the values of the Hilbert function can be recomputed by expanding the rational function (up to each given degree). □

By speaking of the Hilbert function, we honor David Hilbert who verified **the polynomial nature of the Hilbert function** *in the more general setting discussed later in this lecture. This result is contained in the first of Hilbert's two landmark papers in which classical invariant theory culminated (1890, 1893). It is a consequence of the* **syzygy theorem**, *proved in the same paper for that application. But there is much more. In the two papers, Hilbert presented a whole variety of novel ideas whose significance goes far beyond their original application to invariant theory. Besides the two results already mentioned, Hilbert also proved the* **basis theorem**, *the* **Nullstellensatz** *and further fundamental results. These results served as "lemmas" to show that a large class of rings of invariants is finitely generated (see Lecture 8, Section 8.2 for some details), and to give examples which illustrate Hilbert's ideas. In the 1890 paper, Hilbert proved his finiteness result as an application of the basis theorem. Having been criticized for the nonconstructiveness of his proof, he quickly reacted with a second, constructive proof, based on the Nullstellensatz and other results (see Sturmfels (1993) for a modern treatment). His lemmas, however, opened the door to modern abstract algebra. And, they deeply influenced the further development of algebraic geometry.*

Theorem 1.6 (Hilbert's Basis Theorem). *Every ideal of the polynomial ring $K[\boldsymbol{x}] = K[x_1, \dots, x_n]$ is finitely generated.*

Among the different proofs of the basis theorem known today, a proof by Gordan (1899) is of particular interest for us. In this proof, which will be treated later in this lecture, the idea of Gröbner bases made its first appearance.

Remark 1.7. If all ideals of a ring R are finitely generated, then R is said to be a **Noetherian ring**. This is to honor Emmy Noether[1] who showed that these rings are characterized by the ascending chain condition[2] (1921). Note that the condition of being finitely generated and the ascending chain condition carry over from ideals of R to submodules of finitely generated R-modules (see, for instance, Eisenbud (1995), Section 1.4). □

By abuse of notation, the word basis in Hilbert's basis theorem (in Gröbner basis) is used as another name for a (finite) set of generators. Note, however, that an ideal I of a ring R does not admit a basis in the sense of linear algebra (unless I is a principal ideal generated by a nonzerodivisor). In fact, if $r \geq 2$, and if $f_1, \dots, f_r$ are elements of R, there are always nontrivial R-linear combinations of the f_i which are zero:

$$g_1 f_1 + \ldots + g_r f_r = 0 \in R\,. \tag{1.1}$$

For instance, there are always the **Koszul relations** $f_i f_j - f_j f_i = 0$. Hence, $f_1, \dots, f_r$ are not R-linearly independent.

Remark-Definition 1.8. A module over a ring R is **free** if it is zero or if it admits a basis in the usual sense (that is, a set of generators which is R-linearly independent). In these notes, we only consider free R-modules with a finite basis. As in linear algebra, the number of basis elements is, then, independent of the choice of basis. It is called the **rank** of the free module. Given a free R-module F of rank r with a fixed basis, we think of it as the free R-module R^r with its canonical basis (formed by the column vectors ${}^t(1, 0, \dots, 0), \dots, {}^t(0, 0, \dots, 1)$); here, R^r is the direct sum of r copies of R. That is, we consider the elements of F as column vectors with entries in R. □

[1] Emmy Noether was a student of Gordan.

[2] The **ascending chain condition** says that every chain $I_1 \subset I_2 \subset I_3 \subset \dots$ of ideals of R is eventually stationary. That is, $I_m = I_{m+1} = \dots$ for some $m \geq 1$.

We usually think of a relation of type (1.1) as a column vector ${}^t(g_1, \dots, g_r) \in R^r$ and call it a syzygy on $f_1, \dots, f_r$:

Definition 1.9. Let R be a ring, let M be an R-module, and let $f_1, \dots, f_r \in M$. A **syzygy** on $f_1, \dots, f_r$ is an element of the kernel of the homomorphism

$$\varphi : R^r \longrightarrow M\,, \quad \epsilon_i \longmapsto f_i\,,$$

where $\epsilon_1, \dots, \epsilon_r$ is the canonical basis of the free R-module R^r.

If $\ker \varphi$ is finitely generated, we think of the elements of a given finite set of generators for $\ker \varphi$ as the columns of a matrix which we call a **syzygy matrix** of $f_1, \dots, f_r$. □

Exercise 1.10. If $R = K[x, y, z]$, show that

$$\begin{pmatrix} y & z & 0 \\ -x & 0 & z \\ 0 & -x & -y \end{pmatrix}$$

is a syzygy matrix of x, y, z. □

Remark-Definition 1.11 (Free Resolutions). Let R be a Noetherian ring, and let M be a finitely generated R-module.

(1) Choosing a finite set of generators for M corresponds to choosing an epimorphism $\varphi_0 : F_0 = R^{r_0} \to M$ defined as in Definition 1.9. Since R is Noetherian, the kernel of φ_0 is finitely generated, too, and the choice of a finite set of generators corresponds to the choice of an epimorphism $F_1 = R^{r_1} \to \ker \varphi_0$. Taking φ_1 to be the composite $F_1 \to \ker \varphi_0 \subset F_0$, we get a sequence

$$0 \longleftarrow M \xleftarrow{\varphi_0} F_0 \xleftarrow{\varphi_1} F_1$$

which is referred to as a **free presentation** of M. If we think of φ_1 as a matrix (representing the homomorphism with respect to the canonical bases), we call it a **presentation matrix** of M.

(2) Continuing to choose epimorphisms as in (1), we get a sequence

$$0 \longleftarrow M \xleftarrow{\varphi_0} F_0 \xleftarrow{\varphi_1} F_1 \longleftarrow \cdots \longleftarrow F_{i-1} \xleftarrow{\varphi_i} F_i \xleftarrow{\varphi_{i+1}} F_{i+1} \longleftarrow \cdots$$

with free R-modules F_i. The sequence is **exact**, that is, $\operatorname{im} \varphi_{i+1} = \ker \varphi_i$ for all i (in particular, φ_0 is surjective). By abuse of notation, we refer to the sequence, as well as to its “free part”

$$F_0 \xleftarrow{\varphi_1} F_1 \longleftarrow \cdots \longleftarrow F_{i-1} \xleftarrow{\varphi_i} F_i \xleftarrow{\varphi_{i+1}} F_{i+1} \longleftarrow \cdots\,,$$

as a **free resolution** of M. We call $\operatorname{im} \varphi_i$ an ***i*th syzygy module** and its elements ***i*th order syzygies** of M. Note that these modules depend on the choices made, they are determined by M up to free summands only. If we think of φ_i as a matrix, we call it an ***i*th syzygy matrix** of M. Further, we say that the free resolution is **finite** of **length** m if $F_m \neq 0$, but $F_i = 0$ for each $i > m$. Otherwise, the resolution has length ∞. □

In the general situation considered in the definition above, M may not admit a *finite* free resolution (see Lecture 4, Example 4.9). Over the polynomial ring, however, we have the following result:

Theorem 1.12 (Hilbert's Syzygy Theorem). *Every finitely generated $K[x_1, \dots, x_n]$-module has a free resolution of length $\leq n$.*

As for the basis theorem, there are different ways of proving the syzygy theorem. We refer to Remark 1.45 below for a comment on a constructive proof using Gröbner bases.

Free resolutions allow us to deduce information on modules from information on free modules. As a typical example of how this works, we review Hilbert's original application of the syzygy theorem, namely the representation of the infinitely many values of the Hilbert function in finite terms. This requires some preparations.

Remark-Definition 1.13. Let $R = \bigoplus_{\nu \geq 0} R_\nu$ be a graded ring.

(1) If $M = \bigoplus_{\nu \in \mathbb{Z}} M_\nu$ is a graded R-module, and if $d \in \mathbb{Z}$, we denote by $M(d)$ the graded R-module obtained from M by shifting its grading d steps. That is, M and $M(d)$ agree as an R-module, but the grading of $M(d) = \bigoplus_{\nu \in \mathbb{Z}} M(d)_\nu$ is defined by $M(d)_\nu = M_{\nu+d}$. To put it yet in another way, the homogeneous elements of M of degree μ are the homogeneous elements of $M(d)$ of degree $\mu - d$. We refer to $M(d)$ as the **dth twist** of M. In particular, for each d, we have the graded R-module $R(d)$ in which the free generator 1 has degree $-d$. By specifying a basis together with a degree for each basis vector, a free R-module F becomes a **graded free R-module** (with a basis of homogeneous elements). We may then think of F as a direct sum of type $F = \bigoplus_{i=1}^s R(d_i)$. Here, for each i, the ith canonical basis vector of $\bigoplus_{i=1}^s R(d_i)$ is homogeneous of degree $-d_i$.

(2) If M and N are graded R-modules, a **graded homomorphism** from M to N **of degree zero** is a homomorphism $N \leftarrow M$ taking each homogeneous element to a homogeneous element of the same degree. If $F = \bigoplus_{i=1}^s R(d_i)$ and $G = \bigoplus_{j=1}^t R(e_j)$ are graded free R-modules, a graded homomorphism $F \leftarrow G$ of degree 0 is given by an $s \times t$-matrix whose ij entry is a homogeneous element of R of degree $d_i - e_j$, for each pair i, j. We refer to such a matrix as a **graded matrix** over R. □

Example 1.14. If $R = K[w, x, y, z]$, the matrix

$$\varphi = \begin{pmatrix} x+y+z & w^2 - x^2 & z^3 \\ 1 & x & xy + z^2 \end{pmatrix}$$

defines a homomorphism

$$R \oplus R(-1) \xleftarrow{\varphi} R(-1) \oplus R(-2) \oplus R(-3)$$

which is graded of degree zero. Indeed, φ has homogeneous entries of the appropriate degrees. Note that there is a nonzero scalar entry (namely 1).

That is, not all entries of φ are contained in the homogeneous maximal ideal $\langle w, x, y, z \rangle$ of R. $\square$

If M is a module over a ring R, a **minimal set of generators** for M is a set of generators for M such that no proper subset generates M. Examples such as $\{x + x^2, x^2\}$ and $\{x\}$ show that minimal sets of generators for the same module may differ in their number of elements. In the situation considered in the remark below, however, it follows from the graded version of Nakayama's lemma[3] that the number of elements of a minimal set of homogeneous generators for M and their degrees are uniquely determined by M. Note that the remark applies, in particular, to the polynomial ring $K[x_1, \ldots, x_n]$ with its natural grading by degree and its homogeneous maximal ideal $\mathfrak{m} = \langle x_1, \ldots, x_n \rangle$.

Remark-Definition 1.15 (Minimal Free Resolutions). Let $R = \bigoplus_{\nu \geq 0} R_\nu$ be a finitely generated graded $R_0 = K$-algebra. Then R is a Noetherian ring (apply Hilbert's basis theorem as in Eisenbud (1995), Section 1.4). Further,

$$\mathfrak{m} = \bigoplus_{\nu \geq 1} R_\nu$$

is a homogeneous maximal ideal of R containing all proper homogeneous ideals of R. Let M be a finitely generated graded R-module.

(1) The choice of a finite set of *homogeneous* generators $f_1, \ldots, f_{r_0}$ for M corresponds to the choice of a *graded* epimorphism $M \leftarrow F_0 = \bigoplus_{j=1}^{r_0} R(-d_j)$ of degree 0, where $d_j = \deg f_j$. Since R is Noetherian, we may continue to construct a free resolution of M such that all modules are graded, and such that all homomorphisms are graded of degree zero. Such a resolution is called a **graded free resolution** of M. If, at each stage of constructing the resolution, we choose a minimal set of homogeneous generators, we get a **minimal free resolution** of M.

(2) Given a graded free resolution

$$0 \longleftarrow M \xleftarrow{\varphi_0} F_0 \xleftarrow{\varphi_1} F_1 \longleftarrow \ldots \longleftarrow F_{i-1} \xleftarrow{\varphi_i} F_i \xleftarrow{\varphi_{i+1}} F_{i+1} \longleftarrow \ldots,$$

the images of the basis vectors of F_i under φ_i form a minimal set of generators for $\operatorname{im} \varphi_i$ iff $\operatorname{im} \varphi_{i+1} \subset \mathfrak{m} F_i$, that is, iff φ_{i+1}, considered as a graded matrix, does not have a nonzero scalar entry. In fact, the jth row of φ_{i+1} has an entry in $K \setminus \{0\}$ iff the image of the jth basis vector of F_i under φ_i is an R-linear combination of the images of the other basis vectors.

(3) In writing the ith free module F_i of a minimal free resolution of M, we usually collect all copies of R involving the same twist:

[3] We refer to Decker and Schreyer (2006) for Nakayama's lemma and its application in Remarks 1.15 and 1.17.

$$F_i = \bigoplus_j R(-j)^{\beta_{ij}} .$$

Note that, as a consequence of the graded version of Nakayama's lemma, minimal free resolutions are uniquely determined up to a graded **isomorphism of free resolutions**:

$$\begin{tikzcd}
0 & M \arrow[l] \arrow[d, equal] & F_0 \arrow[l] \arrow[d, "\alpha_0"] & F_1 \arrow[l, "\varphi_1"'] \arrow[d, "\alpha_1"] & F_2 \arrow[l, "\varphi_2"'] \arrow[d, "\alpha_2"] & \cdots \arrow[l, "\varphi_3"'] \\
0 & M \arrow[l] & G_0 \arrow[l] & G_1 \arrow[l, "\psi_1"'] & G_2 \arrow[l, "\psi_2"'] & \cdots \arrow[l, "\psi_3"']
\end{tikzcd}$$

That is, the diagram is commutative and the α_i are graded isomorphisms of degree 0. Hence, the **graded Betti numbers** $\beta_{ij}(M) := \beta_{ij}$ and the **graded syzygy modules**

$$\mathrm{Syz}_i(M) := \mathrm{im}\,\varphi_i\,,\ i \geq 1\,,$$

are uniquely determined by M. □

Exercise 1.16. If $R = K[x, y, z]$, show that

$$R \xleftarrow{(x,y,z)} R(-1)^3 \xleftarrow{\begin{pmatrix} y & z & 0 \\ -x & 0 & z \\ 0 & -x & -y \end{pmatrix}} R(-2)^3 \xleftarrow{\begin{pmatrix} z \\ -y \\ x \end{pmatrix}} R(-3) \longleftarrow 0$$

is the minimal free resolution of $R/\langle x, y, z\rangle$. Note that $R/\langle x, y, z\rangle = K$. It is a graded R-module consisting of just one graded piece sitting in degree 0. □

Remark 1.17. A **local ring** is a ring having just one maximal ideal. Due to the local version of Nakayama's lemma, finitely generated modules over local Noetherian rings have uniquely determined minimal free resolutions, too. If such a module M is given, its ***i*th Betti number** is the rank of the ith free module in the minimal free resolution of M. □

In most applications of minimal free resolutions to algebraic geometry, M will be the homogeneous coordinate ring $K[x_0, x_1, \ldots, x_n]/I$ of a projective algebraic set, considered as a module over $K[x_0, x_1, \ldots, x_n]$ (algebraic sets and their coordinate rings will be treated in Lecture 2).

Example 1.18 (Hilbert). The **twisted cubic curve** C in the real projective 3-space $\mathbb{P}^3(\mathbb{R})$ is given parametrically as the image of the map

$$\mathbb{P}^1(\mathbb{R}) \longrightarrow \mathbb{P}^3(\mathbb{R})\,,\ \ (s : t) \longmapsto (s^3 : s^2t : st^2 : t^3)\,.$$

If w, x, y, z are the homogeneous coordinates on $\mathbb{P}^3(\mathbb{R})$, the quadrics

$$f_1 := xz - y^2,\ f_2 := wz - xy,\ f_3 := wy - x^2 \in \mathbb{R}[w, x, y, z] =: R$$

vanish on C. In fact, one can show by elementary means that the polynomials f_1, f_2, f_3 account for all polynomials in R vanishing on C (see Cox, Little, and O'Shea (1997), Chapter 1, §4 and Chapter 8, §4). That is, the ideal $I = \langle f_1, f_2, f_3 \rangle$ generated by f_1, f_2, f_3 in R is the vanishing ideal of C. Thus, $R/I = R/\langle f_1, f_2, f_3 \rangle$ is the homogeneous coordinate ring of C. By "trial and error", we easily find the following two R-linear relations among f_1, f_2, f_3:

$$x \cdot f_1 - y \cdot f_2 + z \cdot f_3 = w \cdot f_1 - x \cdot f_2 + y \cdot f_3 = 0\,.$$

Later in this lecture, we will see that in addition to computing Gröbner bases, Buchberger's algorithm also computes syzygies (successively, it computes free resolutions). For the twisted cubic curve, the algorithm finds the two relations above and shows that they generate all syzygies on f_1, f_2, f_3. In fact, it shows that the minimal free resolution of R/I is as follows (see Exercise 1.49):

$$0 \longleftarrow R/I \longleftarrow R \xleftarrow{\varphi_1} R(-2)^3 \xleftarrow{\varphi_2} R(-3)^2 \longleftarrow 0\,,$$

where

$$\varphi_1 = (xz - y^2, wz - xy, wy - x^2) \text{ and } \varphi_2 = \begin{pmatrix} x & w \\ -y & -x \\ z & y \end{pmatrix}.$$

Recall that a $k \times k$ **minor** of a matrix φ is the determinant of a $k \times k$ submatrix of φ. In our example here, the entries of φ_1, that is, f_1, f_2, f_3, are just the 2×2 minors of φ_2 (with appropriate signs). This is no accident. It is, in fact, a consequence of the theorem of Hilbert-Burch, proved by Hilbert in his 1890 landmark paper to give examples of free resolutions (see Eisenbud (1995), Theorem 20.15). $\square$

From now on, we occasionally present explicit SINGULAR code. A thorough introduction to SINGULAR is postponed to Lecture 3.

Example 1.19. The polynomials defining the twisted cubic curve C have coefficients in $\mathbb{Q}$. This allows us to find the minimal free resolution of its homogeneous coordinate ring R/I by computations over $\mathbb{Q}$ (see Remark 2.9 in Lecture 2). In SINGULAR, the polynomial ring $\mathbb{Q}[w, x, y, z]$ may be defined as follows:

```
> ring R = 0, (w,x,y,z), dp;
```

Note that the input prompt `>` is offered to the user and should not be typed. The `0` refers to $\mathbb{Q}$, the prime field of characteristic zero. The meaning of `dp` will be explained later in this lecture.

```
> ideal I = xz-y2, wz-xy, wy-x2;
> I;            // prints I
I[1]=-y2+xz
I[2]=-xy+wz
I[3]=-x2+wy
> resolution fI = mres(I,0);
```

There are several ways of displaying information on the resolution:

```
> fI;
 1      3      2
R <--  R <--  R
0      1      2
> print(fI[1]);   // the first syzygy matrix of R/I
y2-xz,
xy-wz,
x2-wy
> print(fI[2]);   // the second syzygy matrix of R/I
x, w,
-y,-x,
z, y
> print(betti(fI),"betti");    // the Betti diagram
>                              // (see the remark below)
           0     1     2
------------------------
    0:     1     -     -
    1:     -     3     2
------------------------
total:     1     3     2
```

□

Remark 1.20 (Betti Diagrams). If `I` is a homogeneous ideal of a polynomial ring `R` in `SINGULAR`, and if `fI` is *any* graded free resolution of `R/I` computed with `SINGULAR`, the input line

```
> print(betti(fI),"betti");
```

asks `SINGULAR` to display information on the *minimal* free resolution of `R/I` in form of a diagram which will be referred to as a **Betti diagram**. To understand how to read such a diagram, consider the following example:

```
           0     1     2     3
------------------------------
    0:     1     -     -     -
    1:     -     2     1     -
    2:     -     2     3     1
------------------------------
total:     1     4     4     1
```

A number `i` in the top row refers to the ith free module F_i of the resolution. More precisely, the column with first entry `i` lists the number of free generators of F_i in different degrees and, in the bottom row, the total number of free generators (that is, the rank of F_i). If `k:` is the first entry of a row containing a number β in the column corresponding to F_i, then F_i has β generators in degree $k+i$. That is, β is the graded Betti number $\beta_{ij}(R/I)$ with $j=k+i$. The diagram above indicates, for instance, that F_2 has one generator in degree

3 and three generators in degree 4. In total, the Betti diagram corresponds to a minimal free resolution of type

$$R \longleftarrow R(-2)^2 \oplus R(-3)^2 \longleftarrow R(-3) \oplus R(-4)^3 \longleftarrow R(-5) \longleftarrow 0\,. \qquad \square$$

We are now ready to explain how Hilbert represented the infinitely many values of the Hilbert function in finite terms. To begin with, it is clear from Example 1.5 how to do this for the polynomial ring

$$R = K[\boldsymbol{x}] = K[x_1, \ldots, x_n]$$

and its twists. Namely, in this case,

$$H\big(R(d), \nu\big) = H(R, \nu + d) = \binom{\nu + d + n - 1}{n - 1},$$

which agrees for $\nu \geq -(d + n - 1)$ with the polynomial expression

$$P_{R(d)}(\nu) := \frac{(\nu + d + n - 1)(\nu + d + n - 2) \cdots (\nu + d + 1)}{(n-1)!}\,.$$

Since the Hilbert function is additive with respect to exact sequences of graded modules and graded homomorphisms of degree 0, the syzygy theorem implies the following result (see Eisenbud (1995), Section 1.10):

Theorem 1.21 (Polynomial Nature of Hilbert Functions). *Let M be a finitely generated graded $K[\boldsymbol{x}]$-module. There exists a unique polynomial $P_M(t) \in \mathbb{Q}\,[t]$ such that*

$$H(M, \nu) = P_M(\nu) \textit{ for all } \nu \gg 0\,.$$

This polynomial has degree $\leq n - 1$ and is called the **Hilbert polynomial** *of M.*

In fact, the proof outlined above gives that $H(M, \nu) = P_M(\nu)$ for each

$$\nu \geq \max\{j \mid \beta_{ij}(M) \neq 0 \text{ for some } i\} - n + 1\,.$$

Since

$$H_{K[\boldsymbol{x}]}(t) = \sum_{\nu \in \mathbb{Z}} \binom{\nu + n - 1}{n - 1} \cdot t^\nu = \frac{1}{(1-t)^n}\,,$$

the same proof shows that the Hilbert series $H_M(t)$ is a rational function of t:

Theorem 1.22 (Representation of the Hilbert Series I). *Let M be a finitely generated graded $K[\boldsymbol{x}]$-module with graded Betti numbers β_{ij}. Then*

$$H_M(t) = \frac{\sum_{i,j} (-1)^i \beta_{ij} t^j}{(1-t)^n}\,.$$

Another representation of the Hilbert series as a rational function is obtained as a corollary of Theorem 1.21 (see Bruns and Herzog (1993), Section 4.1):

Theorem 1.23 (Representation of the Hilbert Series II). *Let $M \neq 0$ be a finitely generated graded $K[\boldsymbol{x}]$-module, and let d be the degree of $P_M(t)$. There exists a unique Laurent polynomial $Q_M(t) \in \mathbb{Z}[t, t^{-1}]$ with $Q_M(1) \neq 0$ such that*

$$H_M(t) = \frac{Q_M(t)}{(1-t)^{d+1}}\,.$$

Moreover, if $Q_M(t) = \sum_\nu h_\nu t^\nu$, then $\min\{\nu \mid h_\nu \neq 0\}$ is the least number ν such that $M_\nu \neq 0$.

With notations and assumptions as in the theorem, the Hilbert polynomial P_M can be computed from the Laurent polynomial Q_M: if $Q_M^{(i)}$ denotes the ith formal derivative of Q_M (defined by mimicking the usual rules of differentiation), and if we set

$$a_i = \frac{Q_M^{(i)}(1)}{i!}\,, \quad i = 0, \ldots, d\,,$$

then

$$P_M(t) = \sum_{i=0}^{d} (-1)^{d-i} a_{d-i} \binom{t+i}{i}\,.$$

Moreover, if $Q_M(t) = \sum_{i=\ell}^{r} h_i t^i$ with $h_r \neq 0$, then

$$H(M, r-d-1) \neq P_M(r-d-1)$$

and

$$H(M,\nu) = P_M(\nu) \text{ for each } \nu \geq r-d\,.$$

Remark 1.24. Algebraic geometers use the Hilbert polynomial to rediscover or define numerical invariants of a projective algebraic set and its embedding. For instance, if $I \subset K[\boldsymbol{x}]$ is a homogeneous ideal, then

$$d = \deg P_{K[\boldsymbol{x}]/I} = \dim K[\boldsymbol{x}]/I - 1$$

is the dimension of the projective algebraic set defined by I. We will explain this in Lecture 2. □

Example 1.25. In the example of the twisted cubic curve, we may read off the Hilbert series and the Hilbert polynomial of R/I from its minimal free resolution:

$$H_{R/I}(t) = \frac{1 - 3t^2 + 2t^3}{(1-t)^4} = \frac{1+2t}{(1-t)^2}$$

and

$$P_{R/I}(t) = 3t + 1\,.$$ □

In more complicated examples, computing the Hilbert series via syzygies may be costly. As we will see, the computation of a free resolution of $K[\boldsymbol{x}]/I$ requires the computation of Gröbner bases for I itself and for each kernel needed to construct the free resolution. For the Hilbert series, there is an alternative method which only requires the computation of a Gröbner basis for I. Before explaining all this in detail, we give a SINGULAR example.

Example 1.26. We continue the SINGULAR session from Example 1.19 by computing the Hilbert series of the homogeneous coordinate ring R/I of the twisted cubic curve. To begin with, we use the groebner command to compute a Gröbner basis for I:

```
> ideal GI = groebner(I);
> hilb(GI);
//         1 t^0
//        -3 t^2
//         2 t^3

//         1 t^0
//         2 t^1
// dimension (proj.)  = 1
// degree      = 3
> intvec co1 = hilb(GI,1);
> co1;
1,0,-3,2,0
> intvec co2 = hilb(GI,2);
> co2;
1,2,0
```

We see that the SINGULAR command hilb computes the two representations of the Hilbert series of R/I (and some of the numerical invariants referred to in Remark 1.24). The numerator of each representation is encoded as a vector of type intvec. That is, the entries of the vector are integers in a certain range (see Lecture 3, Remark 3.7).

We make use of the SINGULAR user language to write a procedure for displaying Hilbert polynomials (see Lecture 3, Section 3.8 for more on SINGULAR procedures and Section 3.6 for the built-in command hilbPoly).

```
proc displayHilbPoly(ideal G)
"USAGE:  displayHilbPoly(G), G of type ideal
ASSUME:  G must be a homogeneous Groebner basis for an ideal of the
         active ring in the SINGULAR session; say, G generates the
         homogeneous ideal I of R.
RETURN:  None.
NOTE:    Displays the Hilbert polynomial of R/I.
"
{
  int d = dim(G)-1;        // degree of Hilbert polynomial
  intvec co = hilb(G,2);  // representation II of Hilbert series
```

```
    int s = size(co)-1;      // s = deg(Q_M) +1
    ring Qt = 0, t, dp;      // change active ring to Q[t]
    poly QM = 0;
    for (int i=1; i<=s; i=i+1)
    {
      QM = QM+co[i]*t^(i-1);
    }
    poly QMi = QM;           // the polynomial Q_M(t)
    int ifac = 1;
    list a;
    for (i=1; i<=d+1; i=i+1)
    {
      a = insert(a, subst(QMi,t,1)/ifac, i-1);
      QMi = diff((QMi),t);
      ifac = ifac*i;
    }
    poly PM = (-1)^(d)*a[d+1];
    poly bin = 1;
    for (i=1; i<=d; i=i+1)
    {
      bin = bin*(t+i)/i;     // compute binomial coeff. by recursion
      PM = PM+(-1)^(d-i)*a[d+1-i]*bin;
    }
    print(PM);
  }
```

In our session, having read the procedure into SINGULAR, we apply it as follows:

```
  > displayHilbPoly(GI);
  3t+1
```

□

Gordan, originally one of the major critics of Hilbert's 1890 paper, later on gave his own proof of the basis theorem (1899). His novel idea was to reduce the problem to **monomial ideals**, *that is, to ideals generated by monomials. Gordan's very concise paper consists of four short paragraphs. In the modern terminology that we are about to introduce, he proceeded along the following lines:*

- *The monomials in $K[\boldsymbol{x}]$ may be ordered according to the lexicographic order $>_{lp}$.*
- **Gordan's Lemma.** *Every monomial ideal of $K[\boldsymbol{x}]$ is finitely generated.*
- *Every polynomial $f \in K[\boldsymbol{x}]$ can be written in the form*

$$f = \mathrm{L}(f) + \widetilde{f}\,,$$

with leading term $\mathrm{L}(f) = \mathrm{L}_{>_{lp}}(f)$ and a tail $\widetilde{f}$. If $\mathrm{L}(f)$ is divisible by the leading term $\mathrm{L}(g)$ of another polynomial $g \in K[\boldsymbol{x}]$, then

$$\mathrm{L}(f) >_{\mathrm{lp}} \mathrm{L}\left(f - \frac{\mathrm{L}(f)}{\mathrm{L}(g)} g\right) .$$

- *Every ideal of $K[\boldsymbol{x}]$ has a Gröbner basis with respect to $>_{\mathrm{lp}}$. In particular, every ideal of $K[\boldsymbol{x}]$ is finitely generated.*

Definition 1.27. (1) A **monomial order** on $K[\boldsymbol{x}]$ is a total order $>$ on the set of monomials $\{\boldsymbol{x}^\alpha \mid \alpha \in \mathbb{N}^n\}$ which is multiplicative:

$$\boldsymbol{x}^\alpha > \boldsymbol{x}^\beta \implies \boldsymbol{x}^\gamma \boldsymbol{x}^\alpha > \boldsymbol{x}^\gamma \boldsymbol{x}^\beta \text{ for each } \gamma \in \mathbb{N}^n.$$

(2) Let $>$ be a monomial order on $K[\boldsymbol{x}]$. By abuse of notation, we extend $>$ to terms in $K[\boldsymbol{x}]$. If $a, b \in K$ are nonzero scalars, and if $\boldsymbol{x}^\alpha, \boldsymbol{x}^\beta$ are monomials in $K[\boldsymbol{x}]$ such that $\boldsymbol{x}^\alpha > \boldsymbol{x}^\beta$ (respectively $\boldsymbol{x}^\alpha \geq \boldsymbol{x}^\beta$), we write $a\boldsymbol{x}^\alpha > b\boldsymbol{x}^\beta$ (respectively $a\boldsymbol{x}^\alpha \geq b\boldsymbol{x}^\beta$). This does not give us a partial order on the set of all terms in $K[\boldsymbol{x}]$, but it allows us to compare the terms of any given nonzero polynomial $f \in K[\boldsymbol{x}]$ with each other.

The largest term of f, written $\mathrm{L}(f) = \mathrm{L}_>(f)$, is called the **leading term**, or **initial term**, of f. If $\mathrm{L}(f) = a\boldsymbol{x}^\alpha$, with $a \in K$, then a is called the **leading coefficient** and $\boldsymbol{x}^\alpha$ the **leading monomial** of f. We occasionally refer to $f - \mathrm{L}(f)$ as the **tail** of f, written $\mathrm{tail}(f)$. Further, we set $\mathrm{L}(0) = \mathrm{L}_>(0) = 0$.

(3) A monomial order $>$ on $K[\boldsymbol{x}]$ is

- **global**, if $x_i > 1$ for $i = 1, \dots, n$,
- **local**, if $1 > x_i$ for $i = 1, \dots, n$, and
- **mixed**, otherwise. □

The names global and local come from geometry, referring to the global and local study of an algebraic set. We will deal with local (and mixed) orders in Lecture 9. Now, we mainly focus on global orders. These orders can be characterized as follows:

Remark 1.28. Let $>$ be a monomial order on $K[\boldsymbol{x}]$. The following statements are equivalent:

(1) $>$ is global.

(2) $>$ refines the natural partial order $>_{\mathrm{nat}}$ on $\mathbb{N}^n$. That is,

$$\alpha >_{\mathrm{nat}} \beta \implies \boldsymbol{x}^\alpha > \boldsymbol{x}^\beta,$$

where $\alpha \geq_{\mathrm{nat}} \beta$ iff $\alpha - \beta \in \mathbb{N}^n$ iff $\boldsymbol{x}^\alpha$ is divisible by $\boldsymbol{x}^\beta$.

(3) $>$ is a **well-order**. That is, every nonempty set of monomials in $K[\boldsymbol{x}]$ has a least element with respect to $>$.

The nontrivial implication (2) $\implies$ (3) follows from Gordan's lemma which in this context tells us that every subset of $\mathbb{N}^n$ has at most finitely many minimal elements with respect to $>_{\mathrm{nat}}$. □

Example 1.29. Important global monomial orders on $K[\boldsymbol{x}]$ are:

(1) The **lexicographic order** $>_{\mathtt{lp}}$:

$$\boldsymbol{x}^\alpha >_{\mathtt{lp}} \boldsymbol{x}^\beta \;:\Longleftrightarrow\; \text{the first nonzero entry of } \alpha-\beta \text{ is positive}.$$

In SINGULAR, we write, for instance:

```
> ring R = 0, x(1..7), lp;
```

(2) The **degree reverse lexicographic order** $>_{\mathtt{dp}}$:

$$\boldsymbol{x}^\alpha >_{\mathtt{dp}} \boldsymbol{x}^\beta \;:\Longleftrightarrow \deg \boldsymbol{x}^\alpha > \deg \boldsymbol{x}^\beta \text{ or } (\deg \boldsymbol{x}^\alpha = \deg \boldsymbol{x}^\beta \text{ and the last nonzero entry of } \alpha-\beta \text{ is negative}).$$

In SINGULAR, we write, for instance:

```
> ring R = 0, x(1..7), dp;
```

Note that with respect to both orders, the variables are sorted in the usual way: $x_1 > \cdots > x_n$. □

In the context of computing syzygies, we also need to talk about **monomial orders on free modules**. In what follows, let F be a free $K[\boldsymbol{x}]$-module with a fixed basis $e_1,\ldots,e_s$.

Remark-Definition 1.30. A **monomial** in F is a monomial in $K[\boldsymbol{x}]$ times a basis vector of F, that is, an element of the form $\boldsymbol{x}^\alpha e_i$. A **term** in F is a monomial in F times a scalar, that is, an element of type $a\boldsymbol{x}^\alpha e_i$, where $a\in K$. A submodule of F which is generated by monomials is called a **monomial submodule**. Such a submodule has a uniquely determined **minimal set of monomial generators**. A **monomial order** on F may be defined in the same way as a monomial order on $K[\boldsymbol{x}]$. That is, it is a total order $>$ on the set of monomials in F satisfying

$$\boldsymbol{x}^\alpha e_i > \boldsymbol{x}^\beta e_j \implies \boldsymbol{x}^\gamma\boldsymbol{x}^\alpha e_i > \boldsymbol{x}^\gamma\boldsymbol{x}^\beta e_j \text{ for each } \gamma\in\mathbb{N}^n.$$

In these notes, we require in addition that

$$\boldsymbol{x}^\alpha e_i > \boldsymbol{x}^\beta e_i \iff \boldsymbol{x}^\alpha e_j > \boldsymbol{x}^\beta e_j\,,$$

for all $i,j=1,\ldots,s$. In this way, every monomial order on F induces a unique monomial order on $K[\boldsymbol{x}]$ and notions like global and local carry over to monomial orders on free modules. Finally, given a monomial order on F, we define the **leading term**, the **leading coefficient**, the **leading monomial**, and the **tail** of an element of F as we did for a polynomial in $K[\boldsymbol{x}]$. □

One way of getting a monomial order on F is to pick a monomial order $>$ on $K[\boldsymbol{x}]$, and extend it to F. This can be done in several ways. See Lecture 3, Section 3.2 for a thorough discussion of monomial orders.

Definition 1.31. Let $I \subset F$ be a submodule, and let $>$ be a global monomial order on F.

(1) The **leading module**, or **initial module**, of I with respect to $>$ is the monomial submodule

$$\mathrm{L}(I) := \mathrm{L}_>(I) := \big\langle \mathrm{L}_>(f) \,\big|\, f \in I \big\rangle \subset F\,.$$

That is, $\mathrm{L}(I)$ is generated by the leading terms of the elements of I. If I is an ideal of $K[\boldsymbol{x}]$, we refer to $\mathrm{L}(I)$ as the **leading ideal**, or **initial ideal**, of I.

(2) A finite subset $\mathcal{G} = \{f_1, \dots, f_r\}$ of I is a **Gröbner basis for *I*** with respect to $>$ if

$$\mathrm{L}_>(I) = \big\langle \mathrm{L}_>(f_1), \dots, \mathrm{L}_>(f_r) \big\rangle\,.$$

That is, the leading module of I is generated by the leading terms of the elements of $\mathcal{G}$. We say that a finite subset of F is a **Gröbner basis**, if it is a Gröbner basis for the submodule it generates. □

Remark 1.32. (1) Given a global monomial order $>$ on F, every submodule $I \subset F$ has a Gröbner basis with respect to $>$ (indeed, as pointed out by Gordan, this follows from the fact that $\mathrm{L}_>(I)$ is finitely generated). Every Gröbner basis for I generates I (see Remark 1.40 below).

(2) Leading modules depend on the choice of monomial order. Also, a Gröbner basis with respect to one monomial order may not be a Gröbner basis with respect to another monomial order. □

Macaulay (1927), influenced by the ideas of Hilbert, classified the numerical functions which arise as Hilbert functions of (homogeneous) ideals I. On his way, he showed the following results (see Decker and Schreyer (2006) or Eisenbud (1995) for a modern treatment):

Theorem 1.33 (Macaulay). *Let $I \subset F$ be a submodule, and let $>$ be a global monomial order on F. Then the residue classes of the monomials not in $\mathrm{L}_>(I)$ form a K-vector space basis for F/I.*

Definition 1.34. The monomials not in $\mathrm{L}_>(I)$ are called **standard monomials** (for I, with respect to $>$). □

Theorem 1.35 (Macaulay). *Suppose that F is a graded free $K[\boldsymbol{x}]$-module with a fixed basis of homogeneous elements. Let $I \subset F$ be a graded submodule, and let $>$ be a global monomial order on F. Then the Hilbert function of F/I equals the Hilbert function of $F/\,\mathrm{L}_>(I)$.*

Remark 1.36. (1) The computation of Hilbert functions of monomial submodules is of purely combinatorial nature.

(2) Macaulay's theorems give examples of how $\mathrm{L}_>(I)$ carries information on I itself. As does Gordan's proof of the basis theorem, this shows the use made of Gröbner bases: the key idea is to reduce questions on arbitrary ideals to questions on monomial ideals which are much easier (in fact, often combinatorial in nature). The same idea works for submodules of free modules. □

Hironaka (1964) and, independently, Grauert (1972) introduced **standard bases** *which are to power series rings what Gröbner bases are to polynomial rings. Also, Hironaka and Grauert came up with a division theorem for power series.*

The crucial computational breakthrough, however, is due to Buchberger (1965, 1970). Buchberger's thesis problem, proposed by Gröbner under the influence of Macaulay's work, was to develop tools for computing in quotient rings $K[x_1,\dots,x_n]/I$. Buchberger solved this problem completely using Gröbner bases (it was Buchberger who coined the name Gröbner basis; Gordan's paper was forgotten at that time). Buchberger's ideas enable us, in particular, to compute the standard monomials. The essential new computational ingredient is Buchberger's criterion. This allows us to test whether a given set of generators for an ideal (for a submodule of a free module) is in fact a Gröbner basis for the ideal (the submodule). The test is based on division with remainder in the polynomial ring (in a free module over the polynomial ring). That Buchberger's algorithm can also be used to compute syzygies was observed by Schreyer (1980) and others.

Let F be a free $K[\boldsymbol{x}]$-module with a fixed basis $e_1,\dots,e_s$, and let $>$ be a global monomial order on F.

Notation 1.37. We say that a nonzero term $m = a\boldsymbol{x}^\alpha e_i \in F$ is **divisible** by a nonzero term $m' = b\boldsymbol{x}^\beta e_j \in F$, with **quotient**

$$\frac{m}{m'} := \frac{a}{b}\cdot \boldsymbol{x}^{\alpha-\beta} \in K[\boldsymbol{x}]\,,$$

iff $i = j$ and $\boldsymbol{x}^\alpha$ is divisible by $\boldsymbol{x}^\beta$ (that is, $\alpha \geq_{\text{nat}} \beta$). □

Note that if $I \subset F$ is a monomial submodule, given by monomial generators $m_1,\dots,m_r$, and if $m \in F$ is a further monomial, then $m \in I$ iff m is divisible by at least one of the m_i.

Theorem 1.38 (Division with Remainder). *Let $f_1,\dots,f_r \in F\setminus\{0\}$. For every $f \in F$, there exist $g_1,\dots,g_r \in K[\boldsymbol{x}]$ and an element $h \in F$ such that*

$$f = \sum_{k=1}^{r} g_k f_k + h\,,$$

where:

(DIV 1) $\mathrm{L}(f) \geq \mathrm{L}(g_k f_k)$ *whenever both sides are nonzero.*
(DIV 2') *If h is nonzero, then* $\mathrm{L}(h)$ *is not divisible by any* $\mathrm{L}(f_k)$*; that is,* $\mathrm{L}(h) \notin \langle \mathrm{L}(f_1),\dots,\mathrm{L}(f_r)\rangle$.

Any expression for f as above is called a **standard expression** *for f in terms of the f_k with* **remainder** *h.*

The proof of Theorem 1.38 consists of a **division algorithm** for computing a standard expression. In a single step of the division process, if g is the current dividend, the new dividend is obtained as in Gordan's proof of Hilbert's basis theorem by canceling the leading term: if $\mathrm{L}(g)$ is divisible by $\mathrm{L}(f_k)$ for some k, and if we choose such a k, the new dividend is

$$g - \frac{\mathrm{L}(g)}{\mathrm{L}(f_k)} f_k\,.$$

This process must stop after finitely many steps since $>$ is a well-order and since the leading term of the intermediate dividend decreases at each step.

If instead of just canceling the leading terms we remove arbitrary terms of the intermediate dividends as described above, the algorithm still terminates. Indeed, this follows again from the fact that $>$ is a well-order. As a result of this version of the algorithm, we obtain a standard expression whose remainder h satisfies the following condition:

(DIV 2) If h is nonzero, no term of h is divisible by any $\mathrm{L}(f_k)$.

The division algorithm as described above is indeterminate, in that we allow choices to be made in the computational process, and in that the computed remainder may depend on these choices. We refer to Decker and Schreyer (2006), Chapter 2 for a determinate version of the algorithm.

Example 1.39. The command for division with remainder in `SINGULAR` is `reduce`. Given, for instance, a polynomial f and an ideal I by a set of generators, `reduce(f,I,1);` returns a remainder of f on division by the given generators. The remainder satisfies condition (DIV 2') of Theorem 1.38. Entering `reduce(f,I);` instead, the remainder satisfies the stronger condition (DIV 2) above (the computation might be more expensive in this case). Here is a simple example:

```
> ring R = 0, (x,y), dp;
> ideal I = x2y-y3, x3;
> poly f = x3y+x3;
> reduce(f,I,1);
// ** I is no standard basis
xy3+x3
> reduce(f,I);
// ** I is no standard basis
xy3
```

Note that `SINGULAR` prints the warning

```
// ** I is no standard basis
```

(using `standard basis` as another name for Gröbner basis). To explain why, we write $f_1 = x^2y - y^3$ and $f_2 = x^3$. It is clear from the output above that, in our session, the division algorithm proceeds by first using $\mathrm{L}(f_1) = x^2y$ to cancel $\mathrm{L}(f) = x^3y$. In this way, we get the new dividend $g = xy^3 + x^3$. This dividend is an element of I, but its leading term $\mathrm{L}(g) = xy^3$ is not an element of $\langle \mathrm{L}(f_1), \mathrm{L}(f_2)\rangle$ (in particular, f_1, f_2 do not form a Gröbner basis for I). This is to say that $\mathrm{L}(g)$ is not divisible by any of $\mathrm{L}(f_1), \mathrm{L}(f_2)$ and can therefore not be removed in the division process. As a result, we get a nonzero remainder of f on division by f_1, f_2 though $f \in I = \langle f_1, f_2\rangle$. Thus, the division algorithm shows some undesirable behavior when dividing by the elements of an arbitrary set of generators. Dividing by the elements of a Gröbner basis instead, we get the remainder zero:

```
> ideal GI = groebner(I);
> GI;
GI[1]=x2y-y3
GI[2]=x3
GI[3]=xy3
GI[4]=y5
> reduce(f,GI);
0
```
□

Remark 1.40. (1) If $f = \sum_{k=1}^{r} g_k f_k + h$ is a standard expression for a nonzero $f \in F$ in terms of $f_1, \dots, f_r \in F \setminus \{0\}$, then the leading monomial of f is among the leading monomials of $g_1 f_1, \dots, g_r f_r, h$. It follows that $f_1, \dots, f_r$ form a Gröbner basis iff each element $f \in \langle f_1, \dots, f_r\rangle$ has a standard expression in terms of the f_k with remainder zero (in this case, the remainder in every such standard expression is zero).
(2) Let $I \subset F$ be a submodule, and let $f \in F$. The remainder in a standard expression of f in terms of the elements of a Gröbner basis for I is a **normal form** of f mod I. The normal form only depends on I and $>$ if we require that it satisfies condition (DIV 2) of division with remainder. In this case, the normal form gives the expression for f mod I in terms of the standard monomials. □

If we think of Remark 1.40 (1) as a criterion for $f_1, \dots, f_r$ to be a Gröbner basis, then it is certainly not a practical criterion since it asks us to check a condition for the infinitely many elements in $\langle f_1, \dots, f_r\rangle$. Buchberger's criterion, which we treat next, tells us that it is actually enough to consider finitely many of these elements.

Notation 1.41. (1) If $a\boldsymbol{x}^\alpha e_k$, $b\boldsymbol{x}^\beta e_k$ are two nonzero terms of F involving the same basis element e_k, set

$$\gcd(a\boldsymbol{x}^\alpha e_k, b\boldsymbol{x}^\beta e_k) = x_1^{\min(\alpha_1,\beta_1)} \cdots x_n^{\min(\alpha_n,\beta_n)} e_k \,.$$

(2) Let $f_1, \dots, f_r \in F \setminus \{0\}$. Consider a pair of indices i, j with $i > j$. If $\mathrm{L}(f_i)$ and $\mathrm{L}(f_j)$ involve the same basis element, set

$$m_{ij} = \frac{\mathrm{L}(f_i)}{\gcd(\mathrm{L}(f_i), \mathrm{L}(f_j))} \in K[\boldsymbol{x}]$$

and

$$\mathrm{S}(f_i, f_j) = m_{ji} f_i - m_{ij} f_j \in F\,.$$

If $\mathrm{L}(f_i)$ and $\mathrm{L}(f_j)$ involve different basis elements, set $\mathrm{S}(f_i, f_j) = 0$. □

By abuse of notation, we call $\mathrm{S}(f_i, f_j)$ the **S-polynomial** of f_i and f_j (though, this is a polynomial only in case $F = K[\boldsymbol{x}]$). The S in S-polynomial stands for syzygies. In fact, the S-polynomials are designed to cancel leading terms:

$$m_{ji}\mathrm{L}(f_i) - m_{ij}\mathrm{L}(f_j) = 0\,.$$

Theorem 1.42 (Buchberger's Criterion). *With the notation above, compute for each pair of indices $i > j$ a standard expression for the S-polynomial $\mathrm{S}(f_i, f_j)$ in terms of the f_k with remainder h_{ij}. Then $f_1, \dots, f_r$ form a Gröbner basis iff all h_{ij} are zero.*

We occasionally refer to the computation of the h_{ij} as **Buchberger's test**. If all h_{ij} are zero, the test yields standard expressions of type

$$m_{ji} f_i - m_{ij} f_j = \sum_{k=1}^{r} g_k^{(ij)} f_k\,.$$

These expressions define syzygies on $f_1, \dots, f_r$ which we view as elements

$$G^{(ij)} = {}^t\Big(-g_1^{(ij)}, \dots, (-m_{ij} - g_j^{(ij)}), \dots, (m_{ji} - g_i^{(ij)}), \dots, -g_r^{(ij)}\Big)$$

of the kernel of the map

$$F_0 \longrightarrow F\,, \quad \epsilon_i \longmapsto f_i\,,$$

where F_0 is the free $K[\boldsymbol{x}]$-module $K[\boldsymbol{x}]^r$ with its canonical basis $\epsilon_1, \dots, \epsilon_r$. On F_0, we consider the **induced monomial order** defined as

$$\begin{aligned}\boldsymbol{x}^\alpha \epsilon_i >_0 \boldsymbol{x}^\beta \epsilon_j :\Longleftrightarrow\ & \boldsymbol{x}^\alpha \mathrm{L}(f_i) > \boldsymbol{x}^\beta \mathrm{L}(f_j), \ \text{ or} \\ & \big(\boldsymbol{x}^\alpha \mathrm{L}(f_i) = \boldsymbol{x}^\beta \mathrm{L}(f_j) \ \text{ (up to scalar) and } \ i > j\big).\end{aligned}$$

(Note that the property of being global carries over from $>$ to $>_0$.)

The syzygies $G^{(ij)}$ and the induced monomial order are the key ingredients in Schreyer's proof of Buchberger's criterion (see Schreyer (1991) and Decker and Schreyer (2006)). The same proof yields the following result:

Theorem 1.43 (Schreyer). *Let $f_1, \dots, f_r \in F \setminus \{0\}$ form a Gröbner basis with respect to $>$. The syzygies $G^{(ij)} \in K[\boldsymbol{x}]^r$ arising from Buchberger's test form a Gröbner basis for the syzygies on $f_1, \dots, f_r$ with respect to the induced monomial order on $K[\boldsymbol{x}]^r$. In particular, the $G^{(ij)}$ generate all syzygies on $f_1, \dots, f_r$.*

Remark 1.44. Let $f_1, \dots, f_r \in F \setminus \{0\}$. Buchberger's criterion yields **Buchberger's algorithm** for computing a Gröbner basis for $\langle f_1, \dots, f_r \rangle$:

- Compute the remainders h_{ij} in Buchberger's test. If all h_{ij} are zero, return $f_1, \dots, f_r$.
- If a nonzero remainder h_{ij} occurs, add h_{ij} to the set of generators and start over again.

Note that this algorithm terminates due to the ascending chain condition:

$$\langle \mathrm{L}(f_1), \dots, \mathrm{L}(f_r) \rangle \subsetneq \langle \mathrm{L}(f_1), \dots, \mathrm{L}(f_r), \mathrm{L}(h_{ij}) \rangle \quad \text{if} \quad h_{ij} \neq 0\,.$$

Schreyer's theorem yields **Schreyer's algorithm** for computing a generating set for the syzygies on $f_1, \dots, f_r$ (see Decker and Schreyer (2006)):

- Compute a Gröbner basis $f_1, \dots, f_r, f_{r+1}, \dots, f_{r'}$ with Buchberger's algorithm. On your way, store all syzygies on the elements of the Gröbner basis defined by a standard expression in Buchberger's test; these syzygies generate all syzygies on the elements of the Gröbner basis.
- The syzygies obtained from a division leading to a new generator f_k in Buchberger's test allow us to express f_k in terms of $f_1, \dots, f_{k-1}$. Replacing f_k, $k = r', \dots, r+1$, by this expression in each relation obtained from a division with remainder zero in the test, we get the syzygies on the original generators $f_1, \dots, f_r$. □

Remark 1.45. In theoretical terms, Schreyer's approach to computing syzygies allows one to give a constructive proof of Hilbert's syzygy theorem (see Decker and Schreyer (2006), Chapter 2 or Eisenbud (1995), Chapter 15). □

Example 1.46. Consider $f_1 = xy - y$, $f_2 = -x + y^2 \in K[x,y]$ with the lexicographic order (where $x > y$). Then $\mathrm{L}(f_1) = xy$ and $\mathrm{L}(f_2) = -x$. We compute the standard expression

$$\mathrm{S}(f_2, f_1) = y \cdot f_2 + 1 \cdot f_1 = y^3 - y = 0 \cdot f_1 + 0 \cdot f_2 + y^3 - y\,,$$

and add the nonzero remainder $f_3 := y^3 - y$ to the set of generators. Computing the standard expressions

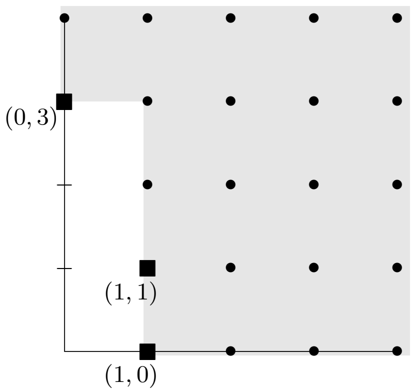
$$\mathrm{S}(f_3, f_1) = x \cdot f_3 - y^2 \cdot f_1 = -xy + y^3 = -1 \cdot f_1 + 0 \cdot f_2 + 1 \cdot f_3$$

and

$$\mathrm{S}(f_3, f_2) = -x \cdot f_3 - y^3 \cdot f_2 = xy - y^5 = 1 \cdot f_1 + 0 \cdot f_2 - (y^2 + 1) \cdot f_3\,,$$

both with remainder zero, we find that f_1, f_2, f_3 form a Gröbner basis for the ideal $I = \langle f_1, f_2 \rangle$.

In the picture on the next page, we visualize the monomials in $K[x,y]$ by printing their exponent vectors. The dots in the shaded region correspond to the monomials in $\mathrm{L}(I)$.



The minimal generators for $\mathrm{L}(I)$ are y^3 and x. By Macaulay's Theorem 1.33, the monomials $1, y, y^2$ not corresponding to dots in the shaded region represent a K-vector space basis for $K[x,y]/I$. Thus, $K[x,y]/I$ is a K-vector space of dimension 3 and every class in $K[x,y]/I$ is represented by a uniquely determined K-linear combination $a + by + cy^2$. To add and multiply classes, we add and multiply the representatives. Hence, the multiplication in $K[x,y]/I$ is determined by the following multiplication table (we write $\overline{f} = f + I$):

$\cdot$	1	$\overline{y}$	$\overline{y}^2$
1	1	$\overline{y}$	$\overline{y}^2$
$\overline{y}$	$\overline{y}$	$\overline{y}^2$	$\overline{y}$
$\overline{y}^2$	$\overline{y}^2$	$\overline{y}$	$\overline{y}^2$

According to Schreyer's algorithm, the matrix

$$\begin{pmatrix} 1 & -y^2+1 & -1 \\ y & 0 & -y^3 \\ -1 & x-1 & -x+y^2+1 \end{pmatrix}$$

is a syzygy matrix of f_1, f_2, f_3. Notice, however, that the third column is superfluous since it is a $K[x,y]$-linear combination of the first two columns. The syzygies on the original generators f_1, f_2 are generated by a single relation, namely the Koszul relation $-f_2 f_1 + f_1 f_2 = 0$. This is obtained by substituting $y \cdot f_2 + 1 \cdot f_1$ for f_3 into the relation given by the second column. $\square$

Remark-Definition 1.47. Let $I \subset F$ be a submodule. A Gröbner basis for I computed with Buchberger's algorithm quite often contains elements whose leading terms are unneeded generators for $\mathrm{L}(I)$. For instance, in Example 1.46, the leading term $\mathrm{L}(f_1) = xy$ is unneeded. By eliminating such generators, by reducing the tail of each remaining generator such that for $i \neq j$ no term of f_i is divisible by $\mathrm{L}(f_j)$, and by adjusting scalars such that each leading coefficient is 1, we obtain the uniquely determined **reduced Gröbner basis** for I. In Example 1.46, the reduced Gröbner basis is $\{x - y^2, y^3 - y\}$. $\square$

Remark 1.48. Buchberger's algorithm generalizes both Gaussian elimination and Euclid's gcd algorithm.

Given linear polynomials

$$f_i = a_{i1}x_1 + \cdots + a_{in}x_n \in K[\boldsymbol{x}] = K[x_1, \ldots, x_n], \quad i = 1, \ldots, r,$$

and a global monomial order $>$ on $K[\boldsymbol{x}]$ such that $x_1 > \cdots > x_n$, computing the reduced Gröbner basis for $\langle f_1, \ldots, f_r \rangle$ amounts to transforming the coefficient matrix $A = (a_{ij})$ into a matrix in reduced row echelon form with pivots 1.

In the case of one variable x, there is precisely one global monomial order: $\cdots > x^2 > x > 1$. If $f_1, f_2 \in K[x]$, the reduced Gröbner basis for $\langle f_1, f_2 \rangle$ with respect to this order consists of exactly one element, namely the greatest common divisor $\gcd(f_1, f_2)$, and Buchberger's algorithm takes the same steps as Euclid's algorithm. □

Exercise 1.49. Compute the minimal free resolution of the homogeneous coordinate ring of the twisted cubic curve and compare your result with Example 1.18. Do not use a computer. □

Exercise 1.50 (Schreyer (1991)). Consider the ideal

$$I = \langle f_1, \ldots, f_5 \rangle \subset R = K[w, x, y, z]$$

generated by the polynomials

$$f_1 = w^2 - xz,\ f_2 = wx - yz,\ f_3 = x^2 - wy,\ f_4 = xy - z^2,\ f_5 = y^2 - wz\,.$$

Using Schreyer's algorithm, compute a graded free resolution of R/I. Is the resolution minimal? □

Remark 1.51. One way of improving Buchberger's algorithm is to reduce the number of S-polynomials to be considered in Buchberger's test. We will discuss this in more detail in Lecture 3, Section 3.5. For one of the more efficient versions of Buchberger's criterion and for a solution of Exercise 1.50 based on that version, see Schreyer (1991) and Decker and Schreyer (2006). □

We finally mention that standard bases of ideals generated by polynomials in formal power series rings can be computed by a variant of Buchberger's algorithm due to Mora (1982) which is based on a different division algorithm. We will discuss this in Lecture 9.

Remark 1.52. Buchberger's algorithm is fundamental to computational algebraic geometry. In fact, all algorithms described in the following lectures are based on Gröbner basis computations (in a few cases, additional techniques are needed).

Quite a number of the algorithms rely on syzygy computations. However, the use made of syzygy computations is not restricted to being part of these algorithms. In fact, when studying geometric objects via their equations, the syzygies often provide additional geometric insight. See Eisenbud (2005) for a book which “illustrates the use of syzygies in many concrete geometric considerations”.

A classical problem to which syzygy computations have been applied successfully in recent years is the classification of smooth surfaces of low degree in projective 4-space (see Decker, Ein, and Schreyer (1993), Decker and Schreyer (2000), Popescu (1993), Popescu and Ranestad (1996), Aure et al (1997), and Abo, Decker, and Sasakura (1998)).

Other applications include the study of the Gale transform by Eisenbud and Popescu (1999, 2000) and Schreyer's approach of using small finite fields to construct special examples of algebraic sets (1996, 2001). □

Remark 1.53 (Further Reading). For more details and complete proofs of the results presented in this lecture, see Bruns and Herzog (1993), Cox, Little, and O'Shea (1997, 1998), Decker and Schreyer (2006), and Eisenbud (1995).

Lecture 2

Basic Computational Problems and Their Solution

Our basic geometric objects of study are sets of solutions of polynomial equations in affine or projective space and are called affine or projective algebraic sets. In this lecture, we introduce the geometry-algebra dictionary which relates algebraic sets to ideals of polynomial rings, translating geometric statements into algebraic statements and vice versa. We pay particular attention to computational problems arising from basic geometric questions. And, we begin to explore how Gröbner bases can be used to solve the problems.

2.1 Computational Problems Arising from the Geometry-Algebra Dictionary

Let K be a field, and let $\mathbb{A}^n(K)$ be the **affine n-space** over K,

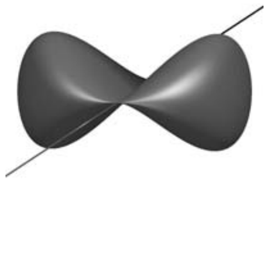
$$\mathbb{A}^n(K) := \big\{(a_1,\dots,a_n) \,\big|\, a_1,\dots,a_n \in K\big\}\,.$$

Each polynomial $f \in K[\boldsymbol{x}] = K[x_1,\dots,x_n]$ defines a function

$$f : \mathbb{A}^n(K) \to K\,,\ (a_1,\dots,a_n) \mapsto f(a_1,\dots,a_n);$$

the value $f(a_1,\dots,a_n)$ is obtained by substituting the a_i for the x_i in f and evaluating the corresponding expression in K. This allows us to talk about the vanishing locus of f in $\mathbb{A}^n(K)$, namely $\mathrm{V}(f) := \{p \in \mathbb{A}^n(K) \mid f(p) = 0\}$. If f is nonconstant, we call $\mathrm{V}(f)$ a **hypersurface** in $\mathbb{A}^n(K)$.

Example 2.1. We visualize the hypersurface $\mathrm{V}\big(y^4 + z^2 - y^2(1-x^2)\big) \subset \mathbb{A}^3(\mathbb{R})$ using `SURF`:



$\square$

A subset $A \subset \mathbb{A}^n(K)$ is an **(affine) algebraic set** if it is the common vanishing locus of finitely many polynomials $f_1, \dots, f_r \in K[\boldsymbol{x}]$:

$$A = \mathrm{V}(f_1, \dots, f_r) := \big\{p \in \mathbb{A}^n(K) \bigm| f_1(p) = \dots = f_r(p) = 0\big\}\,.$$

We then call $f_1 = 0\,, \dots\,, f_r = 0$ a set of **defining equations** for A. Note that every $K[\boldsymbol{x}]$-linear combination $f = \sum_{i=1}^r g_i f_i$ vanishes on A, too. We may, thus, as well say that A is the vanishing locus $\mathrm{V}(I)$ of the ideal $I = \langle f_1, \dots, f_r\rangle$ formed by all these combinations:

$$A = \mathrm{V}(I) := \big\{p \in \mathbb{A}^n(K) \bigm| f(p) = 0 \text{ for all } f \in I\big\}\,.$$

By Hilbert's basis theorem, every ideal I of $K[\boldsymbol{x}]$ is of type $I = \langle f_1, \dots, f_r\rangle$ for some $f_1, \dots, f_r \in K[\boldsymbol{x}]$. Its vanishing locus $\mathrm{V}(I) \subset \mathbb{A}^n(K)$ is, thus, an algebraic set (in fact, the vanishing locus of any given subset of $K[\boldsymbol{x}]$ is defined as above and is an algebraic set). We have $\mathrm{V}(0) = \mathbb{A}^n(K)$, $\mathrm{V}(1) = \emptyset$, $\mathrm{V}(I) \cup \mathrm{V}(J) = \mathrm{V}(I \cap J)$, and $\bigcap_\lambda \mathrm{V}(I_\lambda) = \mathrm{V}(\sum_\lambda I_\lambda)$. In particular, the algebraic subsets of $\mathbb{A}^n(K)$ satisfy the axioms for the closed sets of a topology on $\mathbb{A}^n(K)$. This topology is called the **Zariski topology**. If $X \subset \mathbb{A}^n(K)$ is any subset, then $\overline{X}$ will denote its closure in the Zariski topology.

Having associated an algebraic subset of $\mathbb{A}^n(K)$ to each ideal of $K[\boldsymbol{x}]$, we now proceed in the other direction. Namely, if $A \subset \mathbb{A}^n(K)$ is an algebraic set, we define the **vanishing ideal** of A to be

$$\mathrm{I}(A) = \big\{f \in K[\boldsymbol{x}] \bigm| f(p) = 0 \text{ for all } p \in A\big\}.$$

Vanishing ideals have a property not shared by all ideals: they are radical ideals. Here, if I is an ideal of a ring R, its **radical** is the ideal

$$\sqrt{I} := \big\{f \in R \bigm| f^m \in I \ \text{ for some } \ m \geq 1\big\},$$

and we call I a **radical ideal** if $I = \sqrt{I}$.

Example 2.2. If $f = c \cdot f_1^{m_1} \cdots f_s^{m_s}$ is the factorization of a nonconstant polynomial $f \in K[\boldsymbol{x}]$ into its irreducible coprime factors $f_i \in K[\boldsymbol{x}]$, then

$$\sqrt{\langle f\rangle} = \langle f_1 \cdots f_s\rangle.$$

The product $f_1 \cdots f_s$ is uniquely determined by f up to nonzero scalars and is called the **square-free part** of f. If all m_i are 1, we say that f is **square-free**. In this case, $\langle f\rangle$ is a radical ideal. □

The correspondence between algebraic sets and ideals is made precise by Hilbert's Nullstellensatz which is fundamental to the geometry-algebra dictionary:

Theorem 2.3 (Hilbert's Nullstellensatz). *Let $A \subset \mathbb{A}^n(K)$ be an algebraic set, and let $I \subset K[\boldsymbol{x}]$ be an ideal. If K is algebraically closed, then*

$$A = \mathrm{V}(I) \implies \mathrm{I}(A) = \sqrt{I}.$$

Corollary 2.4. *If K is algebraically closed, then* I *and* V *define a one-to-one correspondence*

$$\{\textit{algebraic subsets of } \mathbb{A}^n(K)\} \underset{\mathrm{V}}{\overset{\mathrm{I}}{\rightleftarrows}} \{\textit{radical ideals of } K[\boldsymbol{x}]\}.$$

Under this correspondence, the points of $\mathbb{A}^n(K)$ correspond to the maximal ideals of $K[\boldsymbol{x}]$.

See, for instance, Decker and Schreyer (2006) for proofs.

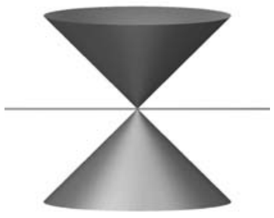
Properties of an ideal I of a ring R can be expressed in terms of the quotient ring R/I. For instance, I is a maximal ideal iff R/I is a field. More generally, I is a prime ideal iff R/I is an integral domain. Finally, I is a radical ideal iff R/I is **reduced** (that is, the only nilpotent element of R/I is zero).

If K is any field, and if $A \subset \mathbb{A}^n(K)$ is an algebraic set, the **coordinate ring** of A is the reduced ring $K[A] := K[\boldsymbol{x}]/\mathrm{I}(A)$. An element of $K[A]$ is, thus, the residue class $\overline{f} = f + \mathrm{I}(A)$ of a polynomial $f \in K[\boldsymbol{x}]$. We may also think of it as a **polynomial function** on A, namely $A \to K$, $p \mapsto f(p)$. The ring $K[A]$ is an integral domain iff A is **irreducible**, that is, iff A cannot be written as the union of two algebraic sets properly contained in A (otherwise, A is **reducible**). If A is irreducible, it is also called an **(affine) variety**. The empty set is not considered to be irreducible.

Example 2.5. (1) The algebraic subset of $\mathbb{A}^3(\mathbb{R})$ with defining equations $x^2z + y^2z - z^3 = x^3 + xy^2 - xz^2 = 0$ is reducible since it decomposes into a cone and a line:

$$\mathrm{V}(x^2z + y^2z - z^3, x^3 + xy^2 - xz^2) = \mathrm{V}(x^2 + y^2 - z^2) \cup \mathrm{V}(x, z)\,.$$

We draw a picture using `SURF`:



(2) The real algebraic set in Example 2.1 is irreducible (even if the picture displayed seems to suggest that it is the union of a surface and a line). See Exercise 2.34 and Lecture 7, Example 7.2. □

If K is any field, and if $I \subset K[\boldsymbol{x}]$ is any ideal, we refer to $K[\boldsymbol{x}]/I$ as an **affine $\boldsymbol{K}$-algebra**, or simply as an **affine ring**. If I is a prime ideal, we refer to $K[\boldsymbol{x}]/I$ as an **affine domain**. Note that every finitely generated K-algebra

S arises as an affine ring. Indeed, choose finitely many generators $s_1, \ldots, s_m$ for S, and represent S as the homomorphic image of the polynomial ring $K[y_1, \ldots, y_m]$ by considering the map $\phi : K[y_1, \ldots, y_m] \twoheadrightarrow S$, $y_i \mapsto s_i$. Then $S \cong K[y_1, \ldots, y_m]/\ker\phi$. If K is algebraically closed, and if S is reduced, then S can be thought of as the coordinate ring of an affine algebraic set. Indeed, if S is reduced, $I := \ker\phi$ is a radical ideal. Hence, $\mathrm{I}(\mathrm{V}(I)) = I$ by the Nullstellensatz, and we may take the algebraic set $\mathrm{V}(I) \subset \mathbb{A}^m(K)$.

Just as affine algebraic sets are given by polynomials, the natural maps between them are also given by polynomials: if $A \subset \mathbb{A}^n(K)$ and $B \subset \mathbb{A}^m(K)$ are algebraic sets, a map $\varphi : A \to B$ is called a **morphism**, or a **polynomial map**, if there exist polynomials $f_1, \ldots, f_m \in K[x_1, \ldots, x_n]$ such that $\varphi(p) = (f_1(p), \ldots, f_m(p))$ for all $p \in A$. In this case, φ has an algebraic counterpart, namely the K-algebra homomorphism

$$\varphi^* : K[B] \longrightarrow K[A],\ \ y_i + \mathrm{I}(B) \longmapsto f_i + \mathrm{I}(A)\,,$$

where $y_1, \ldots, y_m$ are the coordinates on $\mathbb{A}^m(K)$. If we think of the elements of $K[A]$ and $K[B]$ as polynomial functions on A and on B, then φ^* is obtained by composing a polynomial function on B with φ to obtain a polynomial function on A:

$$\begin{tikzcd} A \arrow[rr, "\varphi"] \arrow[dr] & & B \arrow[dl] \\ & K\,. & \end{tikzcd}$$

We are now ready to present a list of basic geometric questions and the algebraic problems arising from them. In formulating the problems, we suppose that I and J are ideals of $K[\boldsymbol{x}]$, each given by a finite set of generators. Our basic reference for the results behind our translation from geometry to algebra and vice versa is Decker and Schreyer (2006).

We begin by recalling that $\mathrm{V}(I) \cap \mathrm{V}(J) = \mathrm{V}(I + J)$. Thus, computing the intersection $\mathrm{V}(I) \cap \mathrm{V}(J)$ just amounts to concatenating the given sets of generators for I and J. It is not immediately clear, however, how to deal with the union of algebraic sets.

- Compute the union $\mathrm{V}(I) \cup \mathrm{V}(J)$. Algebraically, find generators for the intersection $I \cap J$.

The Noetherian property of $K[\boldsymbol{x}]$ implies that every (nonempty) algebraic set can be uniquely written as a finite union $A = V_1 \cup \cdots \cup V_s$ of varieties V_i such that $V_i \not\subset V_j$ for $i \neq j$. The V_i are called the **irreducible components** of A.

- Compute the irreducible components of $\mathrm{V}(I)$. Algebraically, compute the minimal associated primes of I.[1] More generally, compute a primary decomposition of I. More specially, compute the radical of I.

[1] The algebraic problem and the geometric problem described here are only equivalent if K is algebraically closed. See Silhol (1978) for a statement that holds over arbitrary fields.

Remark 2.6 (Primary Decomposition). A proper ideal Q of a ring R is said to be **primary** if $f, g \in R$, $fg \in Q$ and $f \notin Q$ implies $g \in \sqrt{Q}$. In this case, $P = \sqrt{Q}$ is a prime ideal, and Q is also said to be a ***P*-primary ideal**. Given any ideal I of R, a **primary decomposition** of I is an expression of I as an intersection of finitely many primary ideals.

Suppose now that R is Noetherian. Then every proper ideal I of R has a primary decomposition. We can always achieve that such a decomposition $I = \bigcap_{i=1}^{r} Q_i$ is **minimal**. That is, the prime ideals $P_i = \sqrt{Q_i}$ are all distinct and none of the Q_i can be left out. In this case, the P_i are uniquely determined by I and are referred to as the **associated primes** of I. If P_i is minimal among $P_1, \dots, P_r$ with respect to inclusion, it is called a **minimal associated prime** of I. The minimal associated primes of I are precisely the minimal prime ideals containing I. Their intersection is equal to $\sqrt{I}$. Every primary ideal occuring in a minimal primary decomposition of I is called a **primary component** of I. The component is said to be **isolated** if its radical is a minimal associated prime of I. Otherwise, it is said to be **embedded**. The isolated components are uniquely determined by I, the others are far from being unique. See Atiyah and MacDonald (1969) for details and proofs. $\square$

Example 2.7. If $f = c \cdot f_1^{m_1} \cdots f_s^{m_s}$ is the factorization of a nonconstant polynomial $f \in K[\boldsymbol{x}]$ into its irreducible coprime factors $f_i \in K[\boldsymbol{x}]$, then

$$\langle f \rangle = \langle f_1^{m_1} \rangle \cap \ldots \cap \langle f_s^{m_s} \rangle$$

is the (unique) minimal primary decomposition. $\square$

The names isolated and embedded come from geometry. If K is algebraically closed, and if I is an ideal of $R = K[\boldsymbol{x}]$, the minimal associated primes of I correspond to the irreducible components of $\mathrm{V}(I)$, while the other associated primes correspond to irreducible algebraic sets contained (or "embedded") in the irreducible components. A more thorough geometric interpretation of primary decomposition requires the language of schemes (see Eisenbud and Harris (2000) for an introduction to schemes). For instance:

Example 2.8. Let $I = \langle xy, y^2 \rangle \subset K[x, y]$. Then the minimal primary decomposition $I = \langle y \rangle \cap \langle x, y^2 \rangle$ exhibits the affine scheme $X = \mathrm{Spec}(K[x, y]/I)$ as the union of the x-axis and an embedded multiple point at the origin. Observe that there are many different ways of writing X as such a union. For instance, $I = \langle y \rangle \cap \langle x, y \rangle^2$ is a minimal primary decomposition as well. $\square$

In these notes, we will, essentially, avoid to talk about schemes.

Remark 2.9 (The Role of the Coefficient Field). Because of Hilbert's Nullstellensatz, algebraic sets are usually studied over an algebraically closed field such as the field of complex numbers. To visualize geometric objects, however, the field of real numbers is chosen. And, to compute examples with exact computer algebra methods, one typically works over a finite field, the field of rational numbers, or a number field (that is, a finite extension of $\mathbb{Q}$ such as $\mathbb{Q}(\sqrt{2}) \cong \mathbb{Q}[t]/\langle t^2-2\rangle$). See also Remark 4.12 in Lecture 4.

In this context, note that if $K \subset L$ is a field extension, and if I is an ideal of $L[\boldsymbol{x}]$ generated by polynomials $f_1, \dots, f_r$ with coefficients in K, then Buchberger's algorithm applied to $f_1, \dots, f_r$ yields Gröbner basis elements for I which are also defined over K. For almost all geometric questions discussed in this lecture, this allows us to study the vanishing locus of I in $\mathbb{A}^n(L)$ by computations over K.

Note, however, that a prime ideal of $K[\boldsymbol{x}]$ needs not generate a prime ideal of $L[\boldsymbol{x}]$. From a computational point of view, this is reflected by the fact that for computing a primary decomposition, algorithms for polynomial factorization are needed in addition to Gröbner basis techniques (see Lecture 7). In contrast to Buchberger's algorithm, the algorithms for polynomial factorization and their results are highly sensitive to the ground field.

With respect to dimension (see Remark 2.10 below), we point out that if Q is a primary ideal of $K[\boldsymbol{x}]$ with radical P, then the associated primes of $QL[\boldsymbol{x}]$ are precisely the prime ideals of $L[\boldsymbol{x}]$ intersecting $K[\boldsymbol{x}]$ in P and having the same dimension as P (see Zariski and Samuel (1975–1976), Vol II, Chapter VII, §11). See also Section 6.1.1 in Lecture 6.

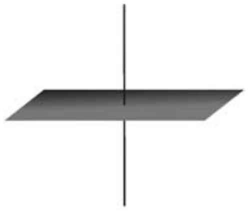
With respect to radicals, note that if K is a perfect field, and if $I \subset K[\boldsymbol{x}]$ is a radical ideal, then also $IL[\boldsymbol{x}]$ is a radical ideal (see again Zariski and Samuel (1975–1976), Vol II, Chapter VII, §11). Recall that finite fields, fields of characteristic zero, and algebraically closed fields are perfect. $\square$

In continuing our problem list, we now present problems for which the geometric interpretation of the algebraic operations under consideration relies on Hilbert's Nullstellensatz. To emphasize this point, we mark such a problem by the square ■ instead of the bullet •.

Convention. *For each problem marked by a square, let $\overline{K}$ be an algebraically closed extension field of K. If $I \subset K[\boldsymbol{x}]$ is an ideal, redefine* $\mathrm{V}(I)$ *to be the vanishing locus of I in $\mathbb{A}^n := \mathbb{A}^n(\overline{K})$. If $A = \mathrm{V}(I) \subset \mathbb{A}^n$, then* $\mathrm{I}(A)$ *is the vanishing ideal of A in $\overline{K}[\boldsymbol{x}]$ and $\overline{K}[A] = \overline{K}[\boldsymbol{x}]/\mathrm{I}(A)$ is its coordinate ring.* $\square$

In what follows, if not otherwise mentioned, I and J are again ideals of $K[\boldsymbol{x}]$, each given by a finite set of generators.

The difference $\mathrm{V}(I) \setminus \mathrm{V}(J)$ need not be an algebraic set. That is, it may not be Zariski closed. As an example, consider the punctured plane obtained by removing the z-axis from the union of the xy-plane and the z-axis.



- Compute the Zariski closure of $\mathrm{V}(I) \setminus \mathrm{V}(J)$. That is, compute the union of those irreducible components of $\mathrm{V}(I)$ which are not contained in $\mathrm{V}(J)$. Algebraically, if I is radical, find generators for the **ideal quotient** of I by J which is defined to be the ideal

$$I : J = \big\{ f \in K[\boldsymbol{x}] \,\big|\, fJ \subset I \big\}.$$

  If I is not necessarily radical, find generators for the **saturation** of I with respect to J, that is, for the ideal

$$\begin{aligned} I : J^\infty &= \big\{ f \in R \,\big|\, fJ^m \subset I \text{ for some } m \geq 1 \big\} \\ &= \bigcup_{m=1}^{\infty} (I : J^m)\,. \end{aligned}$$

- **Solvability and ideal membership.** Decide whether $\mathrm{V}(I)$ is empty. Algebraically, decide whether $1 \in I$. More generally, given any polynomial $f \in K[\boldsymbol{x}]$, decide whether $f \in I$.
- **Radical membership.** Decide whether a given polynomial $f \in K[\boldsymbol{x}]$ vanishes on $\mathrm{V}(I)$. Algebraically, decide whether f is contained in $\sqrt{I}$.
- Compute the **dimension** of $\mathrm{V}(I)$. Algebraically, compute the Krull dimension of the affine ring $K[\boldsymbol{x}]/I$.[2]

The definition of the Krull dimension of a ring is somewhat reminiscent of the fact that the dimension of a vector space over a field is the length of the longest chain of proper subspaces:

Remark 2.10 (Dimension and Codimension). The **Krull dimension** (or simply the **dimension**) of a ring R, denoted $\dim R$, is the supremum of the lengths d of chains

$$P_0 \subsetneq P_1 \subsetneq \ldots \subsetneq P_d$$

of prime ideals of R. If $I \subsetneq R$ is a proper ideal, its **dimension**, written $\dim I$, is defined to be $\dim R/I$. The **codimension** of I, written $\operatorname{codim} I$, is defined as follows. If I is a prime ideal, $\operatorname{codim} I$ is the supremum of lengths of chains of prime ideals with largest ideal $P_d = I$. If I is not necessarily prime, $\operatorname{codim} I$ is the minimum of the codimensions of the prime ideals containing I.

It follows from the definitions that $\dim I + \operatorname{codim} I \leq \dim R$. In general, the inequality may well be strict (see Lecture 9, Example 9.31). Equality holds if R is an affine domain over a field K. In fact, in this case, $\dim R$ equals the transcendence degree of the quotient field of R over K, and this number is the common length of all maximal chains of prime ideals of R (a chain of prime ideals of R is maximal if it cannot be extended to a chain of greater length by inserting a further prime ideal). In particular, $\dim \mathbb{A}^n = \dim K[\boldsymbol{x}] = n$.

[2] The **dimension** of $A = \mathrm{V}(I) \subset \mathbb{A}^n$, written $\dim A$, is defined to be the Krull dimension of the coordinate ring $\overline{K}[A]$. It follows from what we said in Remark 2.9 that $\overline{K}[A]$ and $K[\boldsymbol{x}]/I$ have the same Krull dimension.

An important result in dimension theory, proved using Nakayama's lemma, is **Krull's principal ideal theorem** which asserts that if $I = \langle f \rangle \subsetneq R$ is a principal ideal of a Noetherian ring R and P is a minimal associated prime of I, then $\operatorname{codim} P \leq 1$. If f is a nonzerodivisor of R, each minimal associated prime of $\langle f \rangle$ has precisely codimension 1. In particular, $\dim K[\boldsymbol{x}]/\langle f \rangle = n - 1$ for each nonconstant $f \in K[\boldsymbol{x}]$. In geometric terms, the dimension (of each irreducible component) of a hypersurface $\mathrm{V}(f) \subset \mathbb{A}^n$ is $n-1$.

If $I \subsetneq K[\boldsymbol{x}]$ is any proper ideal, then according to the definition of dimension, $\dim K[\boldsymbol{x}]/I$ is the maximum dimension of a minimal associated prime of I. Geometrically, the dimension of $\mathrm{V}(I)$ in $\mathbb{A}^n$ is the maximum dimension of an irreducible component of $\mathrm{V}(I)$.

See Eisenbud (1995) for details and proofs. □

- Compute the Zariski closure of the image of $\mathrm{V}(I)$ under the projection $\mathbb{A}^n \to \mathbb{A}^{n-k}$ which sends $(a_1, \dots, a_n)$ to $(a_{k+1}, \dots, a_n)$. Algebraically, eliminate the first k variables from I, that is, compute the ***k*th elimination ideal** $I_k = I \cap K[x_{k+1}, \dots, x_n]$.

- More generally, compute the Zariski closure of the image of $\mathrm{V}(I)$ under an arbitrary morphism $\varphi : \mathrm{V}(I) \to \mathbb{A}^m$. Algebraically, if φ is given by polynomials $f_1, \dots, f_m \in K[x_1, \dots, x_n]$, and if $y_1, \dots, y_m$ are the coordinates on $\mathbb{A}^m$, consider the ideal

$$J = IK[\boldsymbol{x}, \boldsymbol{y}] + \big\langle f_1 - y_1, \dots, f_m - y_m \big\rangle \subset K[\boldsymbol{x}, \boldsymbol{y}]\,.$$

Then J defines the graph of φ in $\mathbb{A}^n \times \mathbb{A}^m = \mathbb{A}^{n+m}$, and

$$\overline{\varphi(\mathrm{V}(I))} = \mathrm{V}(J \cap K[\boldsymbol{y}])\,.$$

Remark 2.11. If K is not algebraically closed, and if V_K refers to taking vanishing loci in $\mathbb{A}^n(K)$ and $\mathbb{A}^m(K)$, the Zariski closure of $\varphi(\mathrm{V}_K(I))$ in $\mathbb{A}^m(K)$ may be strictly contained in $\mathrm{V}_K(J \cap K[\boldsymbol{y}])$. Equality holds, however, if $\mathrm{V}_K(I)$ is Zariski dense in the vanishing locus of I in the affine n-space over the algebraic closure of K. Note that if K is infinite, then this condition is fulfilled for $\mathbb{A}^n(K) = \mathrm{V}_K(0)$. Thus, the above applies, in particular, to polynomial parametrizations over infinite fields. Here, a **polynomial parametrization** of an algebraic set $B \subset \mathbb{A}^m(K)$ is a morphism $\varphi : \mathbb{A}^n(K) \to B$ such that $\overline{\varphi(\mathbb{A}^n(K))} = B$. See Decker and Schreyer (2006) for rational parametrizations (and for proofs). □

If a parametrization $\varphi : \mathbb{A}^n(K) \to B$ exists, it allows one to study B in terms of a simpler variety (namely $\mathbb{A}^n(K)$). A more general concept in this direction is normalization. In these notes, we briefly discuss normalization from a computational point of view, addressing the more experienced reader.

- Find the normalization $\widetilde{\mathrm{V}(I)} \to \mathrm{V}(I)$. Algebraically, if I is a radical ideal, find for each (minimal) associated prime P of I the **normalization** of the affine domain $K[\boldsymbol{x}]/P$. That is, find the integral closure of $K[\boldsymbol{x}]/P$ in the quotient field of $K[\boldsymbol{x}]/P$. More precisely, represent the integral closure as an affine domain. As we will see in Lecture 7, this is possible due to a finiteness result of Emmy Noether.

- Check whether $\mathrm{V}(I)$ is smooth. If not, study the behavior of $\mathrm{V}(I)$ at its singular points.

Remark 2.12 (Smooth and Singular Points). Let $A \subset \mathbb{A}^n = \mathbb{A}^n(\overline{K})$ be an algebraic set, and let $p = (a_1, \dots, a_n) \in A$ be a point. We say that A is **smooth** (or **nonsingular**) **at $\boldsymbol{p}$** if the tangent space T_pA to A at p has the expected dimension, that is, if

$$\dim \mathrm{T}_pA = \dim_p A\,.$$

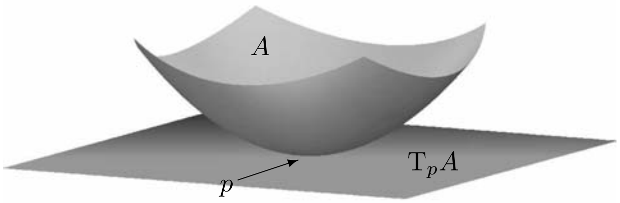
Here, $\dim_p A$ denotes the **dimension of $\boldsymbol{A}$ at $\boldsymbol{p}$**, which is defined to be the maximum dimension of the irreducible components of A through p. Further, the **tangent space to $\boldsymbol{A}$ at $\boldsymbol{p}$** is the linear variety

$$\mathrm{T}_pA = \mathrm{V}(d_pf \mid f \in \mathrm{I}(A)) \subset \mathbb{A}^n\,,$$

where for each $f \in \overline{K}[\boldsymbol{x}]$, we set

$$d_pf = \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}(p)(x_i - a_i) \in \overline{K}[\boldsymbol{x}]\,.$$

This definition extends the concept of tangent spaces from calculus (the partial derivatives are defined in a purely formal way, mimicking the usual rules of differentiation). Note that T_pA is the union of all lines $L = \{p + tv \mid t \in \overline{K}\}$, $v \in \mathbb{A}^n$, such that all polynomials $f(p+tv) \in \overline{K}[t]$, $f \in \mathrm{I}(A)$, vanish with multiplicity ≥ 2 at 0.



If we regard $\mathrm{T}_p\mathbb{A}^n = \mathbb{A}^n$ as an abstract vector space with origin at p and coordinates $X_j = x_j - a_j$, then T_pA is a linear subspace of $\mathrm{T}_p\mathbb{A}^n$. In fact, if $\mathrm{I}(A) = \langle f_1, \dots, f_r \rangle \subset \overline{K}[\boldsymbol{x}]$, then T_pA is the kernel of the linear map $\mathbb{A}^n \to \mathbb{A}^r$ defined by the **Jacobian matrix** at p,

$$\begin{pmatrix} \frac{\partial f_1}{\partial x_1}(p) & \dots & \frac{\partial f_1}{\partial x_n}(p) \\ \vdots & & \vdots \\ \frac{\partial f_r}{\partial x_1}(p) & \dots & \frac{\partial f_r}{\partial x_n}(p) \end{pmatrix}.$$

Note that the definition given above treats T_pA externally, that is, in terms of the ambient space $\mathbb{A}^n$. For an intrinsic definition, consider the **local ring of A at p**,

$$\mathcal{O}_{A,p} = \left\{ \frac{f}{g} \;\middle|\; f, g \in \overline{K}[A],\ g(p) \neq 0 \right\}$$

(formally, this is the localization of $\overline{K}[A]$ at the maximal ideal of all polynomial functions on A vanishing at p). Then $\overline{K}[A] \ni \overline{f} = f + \mathrm{I}(A) \mapsto \mathrm{d}_p f|_{\mathrm{T}_pA}$ induces a natural isomorphism of $\overline{K}$-vector spaces

$$\mathfrak{m}_{A,p}/\mathfrak{m}_{A,p}^2 \xrightarrow{\cong} (\mathrm{T}_pA)^* = \mathrm{Hom}_{\overline{K}}\big(\mathrm{T}_pA, \overline{K}\big)\,,$$

where $\mathfrak{m}_{A,p}$ denotes the unique maximal ideal of $\mathcal{O}_{A,p}$,

$$\mathfrak{m}_{A,p} = \left\{ \frac{f}{g} \;\middle|\; f, g \in \overline{K}[A],\ g(p) \neq 0,\ f(p) = 0 \right\}.$$

On the other hand, making use of the fact that every maximal chain of prime ideals of an affine domain R has length $\dim R$, one can show that

$$\dim_p A = \dim \mathcal{O}_{A,p}\,,$$

and the condition on A to be smooth at p may be expressed intrinsically as

$$\dim_{\overline{K}} \mathfrak{m}_{A,p}/\mathfrak{m}_{A,p}^2 = \dim \mathcal{O}_{A,p}\,.$$

If this holds, we refer to $\mathcal{O}_{A,p}$ as a **regular local ring** (it follows from Krull's principal ideal theorem that we always have $\dim_{\overline{K}} \mathfrak{m}_{A,p}/\mathfrak{m}_{A,p}^2 \geq \dim \mathcal{O}_{A,p}$).

If A is smooth at p, we also say that p is a **smooth** (or **nonsingular**) **point** of A. Otherwise, we say that A is **singular at** p, or that p is a **singular point** of A, or that p is a **singularity** of A. We refer to the set A_{sing} of all singular points of A as the **singular locus** of A. If $A = V_1 \cup \cdots \cup V_s$ is the decomposition of A into its irreducible components, then

$$A_{\text{sing}} = \bigcup_{i \neq j} (V_i \cap V_j) \cup \bigcup_i (V_i)_{\text{sing}}.$$

Starting from this formula, one can show that A_{sing} is an algebraic subset of A such that A and A_{sing} have no irreducible component in common. If A_{sing} is empty, then A is **smooth**. Otherwise, A is **singular**.

See, for instance, Decker and Schreyer (2006) for details and proofs. □

How to compute the singular locus will be discussed in Section 2.2 later in this lecture.

Remark 2.13 (Local Properties). Smoothness of A at p is a **local property** in the sense that it remains unchanged if we replace A by any neighborhood of p in A. Algebraically, this is reflected by the fact that smoothness at

p is expressed in terms of the local ring $\mathcal{O}_{A,p}$. If p is the origin, we may write $\mathcal{O}_{A,p}$ as the quotient $\overline{K}[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle}/\mathrm{I}(A) \cdot \overline{K}[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle}$. Here, if K is any field, we set

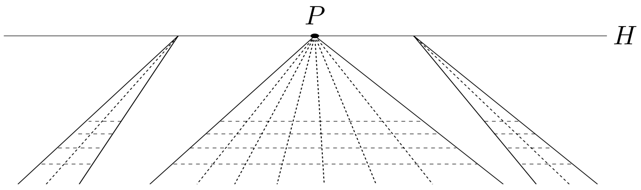
$$K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle} = \left\{ \frac{f}{g} \;\middle|\; f, g \in K[\boldsymbol{x}],\ g \notin \langle \boldsymbol{x} \rangle \right\}.$$

From a computational point of view, we may formulate problems analogous to those discussed so far in this lecture for ideals of $K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle}$ instead of $K[\boldsymbol{x}]$. In Lecture 9, we will give examples of how to interpret these problems geometrically. □

We now turn from the affine to the projective case.

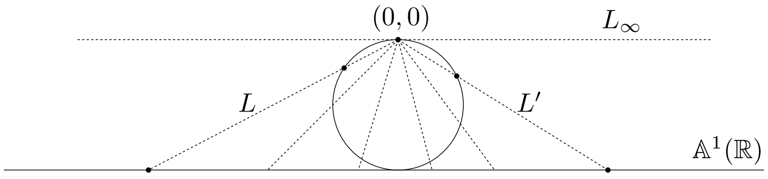
In the affine plane, two lines either meet in a point, or are parallel. In contrast, the projective plane is constructed such that two lines always meet in a point. This is one example of how geometric statements become simpler if we pass from affine to projective geometry.

Historically, the idea of the projective plane goes back to renaissance painters who introduced vanishing points on the horizon to allow for perspective drawing:



We think of a vanishing point (or "point at infinity") as the meeting point of a class of parallel lines in the affine plane $\mathbb{A}^2(\mathbb{R})$. The projective plane $\mathbb{P}^2(\mathbb{R})$ is obtained from $\mathbb{A}^2(\mathbb{R})$ by adding one point at infinity for each such class. A projective line in $\mathbb{P}^2(\mathbb{R})$ is a line $L \subset \mathbb{A}^2(\mathbb{R})$ together with the point at infinity in which the lines parallel to L meet. Further, the horizon, that is, the set of all points at infinity, is a projective line in $\mathbb{P}^2(\mathbb{R})$, the **line at infinity**.

To formalize the idea of the projective plane, we observe that each class of parallel lines in $\mathbb{A}^2(\mathbb{R})$ is represented by a unique line through the origin of $\mathbb{A}^2(\mathbb{R})$. This fits nicely with stereographic projection which allows one to identify the set of lines through the projection center with the real line together with a point at infinity:



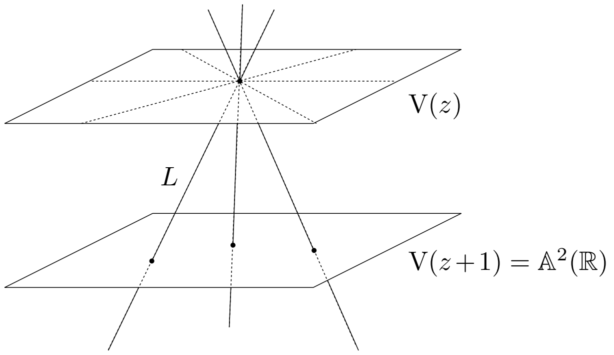
If we define the abstract **projective line** *to be the set*

$$\mathbb{P}^1(\mathbb{R}) = \{\textit{lines through the origin in } \mathbb{A}^2(\mathbb{R})\}$$

and think of it as the line at infinity, we may write

$$\mathbb{P}^2(\mathbb{R}) = \mathbb{A}^2(\mathbb{R}) \cup \mathbb{P}^1(\mathbb{R})\,.$$

Formally, the definition of $\mathbb{P}^2(\mathbb{R})$ is completely analogous to the definition of $\mathbb{P}^1(\mathbb{R})$. To see this, we identify $\mathbb{A}^2(\mathbb{R})$ with the plane $\mathrm{V}(z+1) \subset \mathbb{A}^3(\mathbb{R})$ and $\mathbb{P}^1(\mathbb{R})$ with the set of lines in the xy-plane $\mathrm{V}(z)$ through the origin. Then we may regard $\mathbb{P}^2(\mathbb{R})$ as the set of all lines in $\mathbb{A}^3(\mathbb{R})$ through the origin:



Definition 2.14. *If K is any field, the* **projective n-space over K** *is defined to be the set*

$$\mathbb{P}^n(K) = \{\textit{lines through the origin in } \mathbb{A}^{n+1}(K)\}\,. \qquad \square$$

Each line L through the origin $\mathbf{0} \in \mathbb{A}^{n+1}(K)$ may be represented by a point $(a_0, \dots, a_n) \in L \setminus \{\mathbf{0}\}$. We write $(a_0 : \dots : a_n)$ for the corresponding point of $\mathbb{P}^n(K)$ and call $a_0, \dots, a_n$ a set of **homogeneous coordinates** for this point. Here, the colons indicate that $(a_0, \dots, a_n)$ is determined up to a nonzero scalar multiple only. This representation allows us to think of $\mathbb{P}^n(K)$ as the quotient of $\mathbb{A}^{n+1}(K) \setminus \{\mathbf{0}\}$ modulo the equivalence relation defined by $(a_0, \dots, a_n) \sim (b_0, \dots, b_n)$ iff $(a_0, \dots, a_n) = \lambda(b_0, \dots, b_n)$ for some nonzero scalar λ.

Given a polynomial $f \in K[x_0, \dots, x_n]$, the value $f(a_0, \dots, a_n)$ depends on the choice of representative of the point $p = (a_0 : \dots : a_n) \in \mathbb{P}^n(K)$ and can therefore not be called the value of f at p. Note, however, that if f is **homogeneous**, then $f(\lambda x_0, \dots, \lambda x_n) = \lambda^{\deg(f)} f(x_0, \dots, x_n)$ for all nonzero scalars λ and, thus,

$$f(a_0, \dots, a_n) = 0 \iff \forall\, \lambda \in K \setminus \{0\} :\ f(\lambda a_0, \dots, \lambda a_n) = 0\,.$$

As a consequence, f has a well-defined vanishing locus $\mathrm{V}(f)$ in $\mathbb{P}^n(K)$. If f is nonconstant, we refer to $\mathrm{V}(f)$ as a **hypersurface** in $\mathbb{P}^n(K)$.

A subset $A \subset \mathbb{P}^n(K)$ is a **(projective) algebraic set** if it is the common vanishing locus of finitely many homogeneous polynomials $f_1, \dots, f_r \in K[x_0, \dots, x_n]$:

$$A = \mathrm{V}(f_1, \dots, f_r) := \left\{ p \in \mathbb{A}^n(K) \;\middle|\; f_1(p) = \dots = f_r(p) = 0 \right\}.$$

We then call $f_1 = 0, \dots, f_r = 0$ a set of **defining equations** for A.

If f is a homogeneous linear polynomial, we may identify the algebraic set $\mathrm{V}(f) \subset \mathbb{P}^n(K)$ with $\mathbb{P}^{n-1}(K)$ and its complement $\mathbb{P}^n(K) \setminus \mathrm{V}(f)$ with $\mathbb{A}^n(K)$:

$$\mathbb{P}^n(K) = \mathbb{A}^n(K) \cup \mathbb{P}^{n-1}(K).$$

We refer to $\mathbb{P}^n(K) \setminus \mathrm{V}(f)$ as an **affine chart** of $\mathbb{P}^n$ and to $\mathbb{P}^{n-1}(K)$ as the corresponding **hyperplane at infinity**. For instance, if $f = x_i$, we identify

$$(a_0 : \dots : a_n) \longleftrightarrow \begin{cases} \left(\frac{a_0}{a_i}, \dots, \frac{a_{i-1}}{a_i}, \frac{a_{i+1}}{a_i}, \dots \frac{a_n}{a_i} \right) \in \mathbb{A}^n(K), & \text{if } a_i \neq 0, \\ (a_0 : \dots : a_{i-1} : a_{i+1} : \dots : a_n) \in \mathbb{P}^{n-1}(K), & \text{if } a_i = 0. \end{cases}$$

Remark 2.15 (The Projective Geometry-Algebra Dictionary). The geometry-algebra dictionary relates projective algebraic subsets of projective n-space $\mathbb{P}^n(K)$ to homogeneous ideals of $K[x_0, \dots, x_n]$ via maps I and V essentially defined as in the affine case. If K is algebraically closed, the projective version of the Nullstellensatz implies that there is a one-to-one correspondence

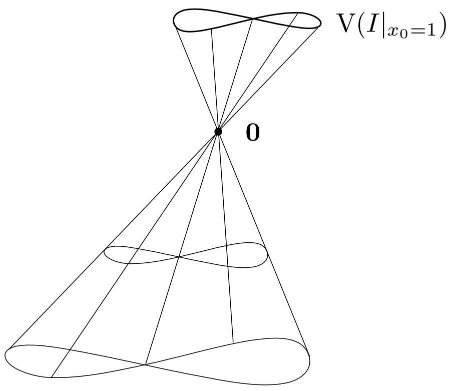
$$\left\{ \begin{matrix} \text{projective algebraic} \\ \text{subsets of } \mathbb{P}^n(K) \end{matrix} \right\} \underset{\mathrm{V}}{\overset{\mathrm{I}}{\rightleftarrows}} \left\{ \begin{matrix} \text{homogeneous radical ideals} \\ \text{of } K[x_0, \dots, x_n] \\ \text{not equal to } \langle x_0, \dots, x_n \rangle \end{matrix} \right\}.$$

Since $\langle x_0, \dots, x_n \rangle$ does not appear in this correspondence, it is called the **irrelevant ideal**.

The **homogeneous coordinate ring** of a projective algebraic subset $A \subset \mathbb{P}^n(K)$ is the reduced graded K-algebra

$$K[A] := K[x_0, \dots, x_n]/\mathrm{I}(A).$$

In terms of affine algebraic sets, this is the coordinate ring of the affine cone over A. Here, if $A \subset \mathbb{P}^n(K)$ is the vanishing locus of a homogeneous ideal $I \subset K[x_0, \dots, x_n]$, the **affine cone** over A is the vanishing locus of I in $\mathbb{A}^{n+1}(K)$:



In the projective case, we may ask questions analogous to those formulated in the affine case. For most of these questions, we consider as in the affine case an algebraically closed extension field $\overline{K}$ of K and redefine $V(I)$ to be the vanishing locus of I in $\mathbb{P}^n := \mathbb{P}^n(\overline{K})$. Further, if $A = V(I) \subset \mathbb{P}^n$, then $I(A)$ is the vanishing ideal of A in $\overline{K}[x_0, \ldots, x_n]$ and $\overline{K}[x_0, \ldots, x_n]/I(A)$ its homogeneous coordinate ring. The computational answers given to our questions in what follows are valid in the projective case as well. Indeed, Buchberger's algorithm applied to homogeneous polynomials yields Gröbner basis elements which are homogeneous, too.

The ideals we are concerned with in explicit computations are often not radical. For instance, if $A = V(I) \subset \mathbb{P}^n(K)$ is a (nonempty) projective algebraic set, given by a homogeneous ideal $I \subset K[x_0, \ldots, x_n]$, then I might have an embedded $\langle x_0, \ldots, x_n\rangle$-primary component (which depends on the choice of a primary decomposition of I). Such a component defines a multiple structure on the vertex of the affine cone over A; it does not contribute to defining A itself. Computing the saturation $I : \langle x_0, \ldots, x_n\rangle^\infty$, we get a simpler ideal defining A. This may not yet be a radical ideal, but at least it does not have an $\langle x_0, \ldots, x_n\rangle$-primary component.

Though we will not need this in what follows, let us mention that there is a one-to-one correspondence between homogeneous ideals I of $K[x_0, \ldots, x_n]$ satisfying $I = I : \langle x_0, \ldots, x_n\rangle^\infty$ and closed subschemes of $\mathbb{P}^n(K)$ (see Hartshorne (1977), Chapter II, Exercise 5.10). □

One further problem arises from adding points at infinity to affine algebraic sets. To describe this problem, we identify $\mathbb{A}^n(K)$ with the affine chart $\mathbb{P}^n(K) \setminus V(x_0)$ of $\mathbb{P}^n(K)$, referring to its complement $V(x_0)$ in $\mathbb{P}^n(K)$ as the **hyperplane at infinity**. Given an affine algebraic set $A \subset \mathbb{A}^n(K)$, we are interested in the **projective closure** of A, which is defined to be the smallest projective algebraic subset of $\mathbb{P}^n(K)$ containing A.

- Given generators for an ideal I of $K[x_1,\dots,x_n]$, compute the projective closure of the affine algebraic set defined by I. Algebraically, compute the **homogenization** I^{hom} of I with respect to the slack variable x_0.

Here, $I^{\text{hom}} \subset K[x_0,\dots,x_n]$ is the ideal generated by the elements

$$f^{\text{hom}} := x_0^{\deg f} \cdot f\left(\frac{x_1}{x_0},\dots,\frac{x_n}{x_0}\right),$$

$f \in I$ (we refer to f^{hom} as the **homogenization of $\boldsymbol{f}$** with respect to x_0).

2.2 Basic Applications of Gröbner Bases

All problems posed in the preceeding section can be settled using Gröbner basis techniques (for radicals and primary decomposition, additional techniques are needed). How to compute in the local ring $K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle}$, for instance, will be explained in Lecture 9. For radicals and primary decomposition, we refer to Lecture 7. In the same lecture, we will discuss normalization. Solutions to the other problems will be provided in this section (see Lecture 3, Section 3.6 for a detailed discussion of the corresponding `SINGULAR` commands).

To begin with, observe that Remark 1.40 settles the **ideal membership problem**. More generally, it settles the submodule membership problem:

Problem 2.16 (Submodule Membership). Given a free $K[\boldsymbol{x}]$-module F with a fixed basis and nonzero elements $f,\ f_1,\dots,f_r \in F$, decide whether

$$f \in I := \langle f_1,\dots,f_r \rangle \subset F.$$

[If so, express f as a $K[\boldsymbol{x}]$-linear combination $f = g_1 f_1 + \ldots + g_r f_r$.]

Solution. Compute a Gröbner basis $f_1,\dots,f_r,f_{r+1},\dots,f_{r'}$ for I using Buchberger's algorithm and a standard expression for f in terms of $f_1,\dots,f_{r'}$ with remainder h. If $h = 0$, then $f \in I$. [In this case, for $k = r',\dots,r+1$, successively do the following: in the standard expression, replace f_k by the expression for f_k in terms of $f_1,\dots,f_{k-1}$ given by the syzygy leading to f_k in Buchberger's test (this requires the relevant syzygies to be stored during Buchberger's test).] □

Example 2.17. Consider the lexicographic Gröbner basis

$$f_1 = xy - y\,,\quad f_2 = -x + y^2,\quad f_3 = y^3 - y$$

for the ideal $I = \langle f_1, f_2 \rangle$ of $K[x,y]$ computed in Lecture 1, Example 1.46. Let

$$f = x^2y - xy + y^3 - y\,.$$

Then $f = x \cdot f_1 + 1 \cdot f_3$ is a standard expression for f in terms of f_1, f_2, f_3 with remainder 0, so $f \in I$. Reconsidering the computation in Example 1.46, we see that we have to substitute $y \cdot f_2 + 1 \cdot f_1$ for f_3 in the standard expression. This gives

$$f = (x+1) \cdot f_1 + y \cdot f_2. \qquad \square$$

As already explained, **solvability** can be decided via ideal membership. Similarly for **radical membership**: if an ideal $I \subset K[\boldsymbol{x}]$ and a polynomial $f \in K[\boldsymbol{x}]$ are given, then

$$f \in \sqrt{I} \iff 1 \in \langle I, tf - 1 \rangle \subset K[\boldsymbol{x}, t],$$

where t is a slack variable.

One way of computing **intersections of ideals** and **ideal quotients** also asks for involving rings with extra variables (see Cox, Little, and O'Shea (1997)). Alternatively, proceed as follows. Given ideals $I = \langle f_1, \ldots, f_r \rangle$ and $J = \langle g_1, \ldots, g_s \rangle$ of $K[\boldsymbol{x}]$, compute the syzygies on the columns of the matrix

$$\begin{pmatrix} 1 & f_1 & \ldots & f_r & 0 & \ldots & 0 \\ 1 & 0 & \ldots & 0 & g_1 & \ldots & g_s \end{pmatrix}.$$

The entries of the first row of the resulting syzygy matrix generate $I \cap J$. In the same way, we obtain a generating set for the ideal quotient

$$I : J = \big\{ f \in K[\boldsymbol{x}] \bigm| fJ \subset I \big\}$$

from the matrix

$$\begin{pmatrix} g_1 & f_1 & \ldots & f_r & 0 & \ldots & & & \ldots & 0 \\ g_2 & 0 & \ldots & 0 & f_1 & \ldots & f_r & 0 & \ldots & 0 \\ \vdots & & & & & \ddots & & & & \\ g_s & 0 & \ldots & & & \ldots & 0 & f_1 & \ldots & f_r \end{pmatrix}.$$

Note that the **intersection of** two **submodules** I, J of a free $K[\boldsymbol{x}]$-module F and their **submodule quotient** $I : J = \big\{ f \in K[\boldsymbol{x}] \bigm| fJ \subset I \big\} \subset K[\boldsymbol{x}]$ are obtained by similar recipes.

Since $I : J^m = (I : J^{m-1}) : J$, the **saturation**

$$I : J^\infty = \bigcup_{m=1}^{\infty} (I : J^m)$$

can be computed by iteration. Indeed, the ascending chain

$$I : J \subset I : J^2 \subset \cdots \subset I : J^m \subset \ldots$$

is eventually stationary since $K[\boldsymbol{x}]$ is Noetherian.

If $K[\boldsymbol{x}]/I$ is a graded affine ring, we already know that its **Hilbert series** $H_{K[\boldsymbol{x}]/I}(t)$ and its **Hilbert polynomial** $P_{K[\boldsymbol{x}]/I}$ can be computed via Gröbner bases (see Macaulay's Theorem 1.35 and Remark 1.36 in Lecture 1). This gives us one way of computing the dimension of homogeneous ideals. Indeed,

$$\dim K[\boldsymbol{x}]/I = \deg P_{K[\boldsymbol{x}]/I} + 1 \tag{2.1}$$

(see Bruns and Herzog (1993) or Eisenbud (1995)).

Remark-Definition 2.18. In algebraic geometry, we make use of the Hilbert polynomial to define or rediscover **numerical invariants** of a projective algebraic set and its embedding. For this purpose, if $A \subset \mathbb{P}^n$ is a projective algebraic set with homogeneous coordinate ring $\overline{K}[A] = \overline{K}[x_0, \ldots, x_n]/\mathrm{I}(A)$, we define the **Hilbert polynomial of A** to be the polynomial $P_A(t) = P_{\overline{K}[A]}(t)$. If d is the degree of $P_A(t)$, the Krull dimension of $\overline{K}[A]$ equals $d+1$ (see equation (2.1) above). In geometric terms, the dimension of the affine cone over A is $d+1$. The **dimension** of A itself is defined to be $\dim A = d$. The **degree** of A is defined to be $d!$ times the leading coefficient of $P_A(t)$. Geometrically, the degree of A is the number of points in which A meets a sufficiently general linear subspace of $\mathbb{P}^n$ of complementary dimension $n-d$ (see, for instance, Decker and Schreyer (2006)). Further, the **arithmetic genus** of A is defined to be $p_a(A) = (-1)^d\big(P_A(0) - 1\big)$. □

Example 2.19. The Hilbert polynomial of the twisted cubic curve $C \subset \mathbb{P}^3$ is $P_C(t) = 3t+1$ (see Lecture 1, Example 1.25). In particular, C has dimension 1 and degree 3. This justifies the name cubic curve. Moreover, C has arithmetic genus 0. □

In Lecture 6, Section 6.1.1, we will discuss an alternative way of computing dimension which applies to nonhomogeneous ideals, too. As for the Hilbert polynomial, Gröbner bases are used to reduce the general problem to a problem concerning monomial ideals. The `SINGULAR` command `dim` is based on this approach.

We now turn to the computation of the singular locus of an algebraic set. For this, we need the following notation. If $I \subsetneq K[\boldsymbol{x}]$ is a proper ideal, we say that I has **pure codimension** c if all its minimal associated primes have codimension c. Also, I is called **unmixed** if it has no embedded components. In many cases of interest, the following criterion allows one to compute the singular locus (and to check that the given ideal is radical):

Theorem 2.20 (Jacobian Criterion). *Let K be a field with algebraically closed extension field $\overline{K}$, let $I = \langle f_1, \ldots, f_r\rangle \subsetneq K[\boldsymbol{x}]$ be an ideal of pure codimension c, and let $A = \mathrm{V}(I)$ be the vanishing locus of I in $\mathbb{A}^n = \mathbb{A}^n(\overline{K})$. If $J \subset K[\boldsymbol{x}]$ is the ideal generated by the $c \times c$ minors of the Jacobian matrix*

$$\begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_r}{\partial x_1} & \cdots & \frac{\partial f_r}{\partial x_n} \end{pmatrix},$$

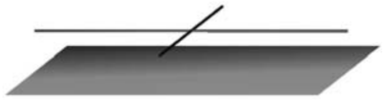
then:

(1) The vanishing locus of $J + I$ in $\mathbb{A}^n$ contains the singular locus A_{sing}.
(2) If $1 \in J + I$, then A is smooth and $I\,\overline{K}[\boldsymbol{x}] = \mathrm{I}(A)$.
(3) If $1 \notin J + I$, suppose in addition that I is unmixed (altogether, we ask that all associated primes of I are minimal and of codimension c). If $\mathrm{codim}(J+I) > \mathrm{codim}\, I$, then $\mathrm{V}(J+I) = A_{\mathrm{sing}}$ and $I\,\overline{K}[\boldsymbol{x}] = \mathrm{I}(A)$.

See Decker and Schreyer (2006), Chapter 4 for a proof and Eisenbud (1995), Section 16.6 and Exercise 11.10 for an algebraic version of the criterion.

The following examples show that the assumptions made in the Jacobian criterion are really needed (in each example, J denotes the respective ideal defined in the criterion).

Example 2.21. (1) Let $I \subset K[x, y, z]$ be the ideal generated by $f_1 = z^2 - z$ and $f_2 = xyz$. Then $I = \langle z\rangle \cap \langle z-1, x\rangle \cap \langle z-1, y\rangle$ has codimension 1, but is not of pure codimension. We have $1 = (2z-1)\frac{\partial f_1}{\partial z} - 4f_1 \in J + I$. However, $A = \mathrm{V}(I) \subset \mathbb{A}^3$ is not smooth. In fact, A is the union of a plane and a pair of lines intersecting in a point which is necessarily a singular point of A:



(2) Applying the Jacobian criterion to the mixed ideal $I = \langle xy, y^2\rangle$, we get $J + I = \langle x, y\rangle$. In contrast, the x-axis $\mathrm{V}(I) \subset \mathbb{A}^2$ is smooth.

(3) The ideal $I = \langle xy^2\rangle \subset K[x, y]$ is unmixed and of pure codimension 1. Its vanishing locus $A = \mathrm{V}(I) \subset \mathbb{A}^2$ is the union of the coordinate axes. Thus, A is singular precisely at the origin. In contrast, the algebraic set defined by the ideal $J + I = \langle xy, y^2\rangle$ in $\mathbb{A}^2$ is the whole x-axis (note that $\operatorname{codim}(J + I) = \operatorname{codim} I = 1$). Scheme-theoretically, I defines the y-axis together with the x-axis doubled. □

Remark 2.22. If $I \subsetneq K[\boldsymbol{x}]$ is any proper ideal, the singular points of $\mathrm{V}(I) \subset \mathbb{A}^n$ arise as the singular points of each irreducible component of $\mathrm{V}(I)$ together with the points of intersection of any two of the components (see Remark 2.12). Thus, if the Jacobian criterion does not apply directly to I, we can combine it with some of the more expensive decomposition techniques discussed in Lecture 7. Indeed, since the Jacobian criterion applies (in particular) to prime ideals, and since we already know how to compute the sum and intersection of ideals, the computation of the singular locus of $\mathrm{V}(I)$ can be reduced to the computation of the minimal associated primes of I. Alternatively, compute an equidimensional decomposition of the radical of I first. □

Remark 2.23 (Jacobian Criterion in the Projective Case). Let I be a proper homogeneous ideal of $K[x_0, \dots, x_n]$, and let $A = \mathrm{V}(I)$ be the vanishing locus of I in $\mathbb{P}^n = \mathbb{P}^n(\overline{K})$. Suppose that I is of pure codimension c and let J be the corresponding ideal of minors as in Theorem 2.20. Further, suppose that $1 \notin J + I$ (otherwise, A is a linear subspace of $\mathbb{P}^n$). Applying Theorem 2.20 to the affine cone over A, we get:

(1) If $\operatorname{codim}(J + I) = n + 1$, then A is smooth.

(2) If all associated primes of I are minimal and of codimension c, and if $\operatorname{codim}(J + I) > \operatorname{codim} I$, then $J + I$ defines the singular locus of A.

Remark 2.22 applies accordingly. □

Determinantal ideals of "expected" codimension provide interesting examples of ideals which are unmixed and pure codimensional (for instance, consider the ideal defining the twisted cubic curve). To state a precise result, let M be a $p \times q$ matrix with entries in $K[\boldsymbol{x}]$, and let $I_k(M) \subset K[\boldsymbol{x}]$ be the ideal generated by the $k \times k$ minors of M, for some p, q, k. Suppose that $I_k(M)$ is a proper ideal of $K[\boldsymbol{x}]$.

Proposition 2.24. *The codimension of every minimal associated prime of $I_k(M)$ and, thus, of $I_k(M)$ itself is at most $(p-k+1)(q-k+1)$.*

Theorem 2.25. *If the codimension of $I_k(M)$ is exactly $(p-k+1)(q-k+1)$, then $K[\boldsymbol{x}]/I_k(M)$ is a Cohen-Macaulay ring.*

We will study Cohen-Macaulay rings in Lecture 5. In this section, we need the corollary below which follows from Theorem 2.25 by applying Theorem 5.41 to the zero ideal of $K[\boldsymbol{x}]/I_k(M)$.

Corollary 2.26 (Unmixedness Theorem). *If the codimension of $I_k(M)$ is exactly $(p-k+1)(q-k+1)$, then all associated primes of $I_k(M)$ are minimal and have this codimension.*

We refer to Eisenbud (1995), Section 18.5 and the references cited there for details and proofs. See also Arbarello et al (1985), Chapter II.

Example 2.27. For the following computation in SINGULAR, we choose $K = \mathbb{Q}$ as our coefficient field. In our geometric interpretation, however, we deal with curves in $\mathbb{P}^3 = \mathbb{P}^3(\mathbb{C})$.

To begin with, we define a ring R implementing $\mathbb{Q}[x_0, \ldots, x_3]$ and a 4×1 matrix A with entries in R:

```
> ring R = 0, x(0..3), dp;
> matrix A[4][1] = x(0),x(1),0,0;
```

Next, we randomly create a 4×2 matrix of linear forms in R. For this, we load the SINGULAR library random.lib and use its command randommat (see Lecture 3 for libraries):

```
> LIB "random.lib";  // loads other libraries incl. matrix.lib
>                    // and elim.lib, too
> matrix B = randommat(4,2,maxideal(1),100);
```

(note that maxideal(k) returns the monomial generators for the k-th power of the homogeneous maximal ideal of the ring R). Concatenating the matrices B and A, we get a 4×3 matrix M of linear forms:

```
> matrix M = concat(B,A);  // from matrix.lib
> print(M);
10*x(0)+62*x(1)-33*x(2)+26*x(3), 42*x(0)-12*x(1)-26*x(2)-65*x(3), x(0),
98*x(0)+71*x(1)+36*x(2)+79*x(3), 22*x(0)+84*x(1)-8*x(2)-55*x(3),  x(1),
-82*x(0)-8*x(1)+33*x(2)+56*x(3), -29*x(0)+43*x(1)+46*x(2)+57*x(3),0,
-60*x(0)+60*x(1)-90*x(2)-78*x(3),37*x(0)+93*x(1)+100*x(2)-50*x(3),0
```

We create the ideal I which is generated by the maximal minors of M and compute its codimension (applied to a ring R, the SINGULAR command nvars returns the number of variables in R):

```
> ideal I = minor(M,3);
> ideal GI = groebner(I);
> int codimI = nvars(R) - dim(GI); codimI;
2
```

So I has the expected codimension $2 = (4-3+1)(3-3+1)$. It is, thus, unmixed and of pure codimension 2 by Corollary 2.26. We check that the assumption on the codimension in the Jacobian criterion is satisfied:

```
> ideal singI = groebner(minor(jacob(GI),codimI) + I);
> nvars(R) - dim(singI);
3
```

Applying the Jacobian criterion and summing up, we see that the vanishing locus C of I in $\mathbb{P}^3$ is a curve, that I generates the vanishing ideal of C in $\mathbb{C}[x_0,\dots,x_3]$, and that singI defines the singular locus of C. We visualize the number of generators of singI and their degrees by displaying the Betti diagram of singI (see Remarks 1.20 and 3.34 for the betti command):

```
> print(betti(singI,0),"betti");
           0     1
------------------
    0:     1     -
    1:     -     -
    2:     -     4
    3:     -    20
------------------
total:     1    24
```

As it turns out, singI comes with an $\langle x_0,\dots,x_3\rangle$-primary component. We get rid of this component by saturating singI with respect to $\langle x_0,\dots,x_3\rangle$:

```
> ideal singI_sat = sat(singI,maxideal(1))[1]; // from elim.lib
> print(betti(singI_sat,0),"betti");
           0     1
------------------
    0:     1     2
    1:     -     1
------------------
total:     1     3
> singI_sat;
singI_sat[1]=x(1)
singI_sat[2]=x(0)
singI_sat[3]=3297*x(2)^2-2680*x(2)*x(3)-5023*x(3)^2
```

We read from the output that C has two singular points which lie on the line $L = \mathrm{V}(x_0, x_1)$. In fact, L is a component of C. We check this via ideal membership (see Problem 2.16 and Lecture 3, Section 3.6.1):

```
> ideal IL = x(0),x(1);
> reduce(I,groebner(IL),1);
_[1]=0
_[2]=0
_[3]=0
_[4]=0
```

By saturating with respect to `IL`, we get an ideal defining the components of C other than L (in fact, since `I` is radical, this amounts to just one ideal quotient computation):

```
> ideal I' = sat(I,IL)[1];  // result is a Groebner basis
> degree(GI);
// dimension (proj.)  = 1
// degree (proj.)   = 6
> degree(I');
// dimension (proj.)  = 1
// degree (proj.)   = 5
```

Since `I` is a radical ideal of pure codimension 2, the same holds for `I'` (in fact, `I'` is the intersection of the (minimal) associated primes of `I` other than $\langle x_0, x_1\rangle$). We may, thus, use the Jacobian criterion to check that C' is smooth:

```
> int codimI' = nvars(R)-dim(I');
> ideal singI' = minor(jacob(I'),codimI') + I';
> nvars(R) - dim(groebner(singI'));
4
```

Since C' and L are smooth, the two singular points of $C = C' \cup L$ must arise as intersection points of C' and L. Thus, L is a secant line to C'. □

Buchberger's algorithm requires the choice of a global monomial order. Its performance and the resulting Gröbner basis depend on the chosen order. For the type of computations done so far in these lectures, in principle any Gröbner basis and, thus, any global monomial order will do. *With respect to efficiency, however, the degree reverse lexicographic order is usually preferable* (see Bayer and Stillman (1987) for some remarks in this direction).

The applications discussed next rely on Gröbner bases whose computation requires the choice of special monomial orders.

Elimination. Let $\boldsymbol{s} \subset \boldsymbol{x} = \{x_1, \dots, x_n\}$ be a subset of variables, and let I be an ideal of $K[\boldsymbol{x}]$. We explain how to eliminate the variables in $\boldsymbol{s}$ from I, that is, how to compute the **elimination ideal** $I \cap K[\boldsymbol{x} \setminus \boldsymbol{s}]$.

Definition 2.28. A monomial order $>$ on $K[\boldsymbol{x}]$ is called an **elimination order** with respect to $\boldsymbol{s}$ (the variables in $\boldsymbol{s}$) if the following implication holds for all $f \in K[\boldsymbol{x}]$:

$$\mathrm{L}(f) \in K[\boldsymbol{x}\setminus\boldsymbol{s}] \implies f \in K[\boldsymbol{x}\setminus\boldsymbol{s}]\,.$$

In this case, we also say that $>$ has the **elimination property** with respect to $\boldsymbol{s}$ (the variables in $\boldsymbol{s}$). □

Example 2.29. Let $\boldsymbol{s} \subset \boldsymbol{x}$, and let $\boldsymbol{t} := \boldsymbol{x}\setminus\boldsymbol{s}$. Moreover, let $>_{\boldsymbol{s}}$ on $K[\boldsymbol{s}]$ and $>_{\boldsymbol{t}}$ on $K[\boldsymbol{t}]$ be monomial orders. The **product order** (or **block order**) $> = (>_{\boldsymbol{s}}, >_{\boldsymbol{t}})$ on $K[\boldsymbol{x}]$ is defined by

$$\boldsymbol{s}^\alpha\boldsymbol{t}^\gamma > \boldsymbol{s}^\beta\boldsymbol{t}^\delta \;:\!\iff\; \boldsymbol{s}^\alpha >_{\boldsymbol{s}} \boldsymbol{s}^\beta \;\text{ or }\; (\boldsymbol{s}^\alpha = \boldsymbol{s}^\beta \;\text{ and }\; \boldsymbol{t}^\gamma >_{\boldsymbol{t}} \boldsymbol{t}^\delta).$$

It is a monomial order which has the elimination property with respect to $\boldsymbol{s}$ iff $>_{\boldsymbol{s}}$ is global, and which is global iff $>_{\boldsymbol{s}}$ and $>_{\boldsymbol{t}}$ are global. A particular example of a product order is the lexicographic order on $K[\boldsymbol{x}]$ which is an elimination order with respect to *each initial set of variables* $\boldsymbol{s} = \{x_1, \dots, x_k\}$, $k = 1, \dots, n$. □

Proposition 2.30. *Let $>$ be a global elimination order on $K[\boldsymbol{x}]$ with respect to $\boldsymbol{s} \subset \boldsymbol{x}$, and let $\mathcal{G}$ be a Gröbner basis for I with respect to $>$. Then $\mathcal{G} \cap K[\boldsymbol{x}\setminus\boldsymbol{s}]$ is a Gröbner basis for $I \cap K[\boldsymbol{x}\setminus\boldsymbol{s}]$ with respect to the restriction of $>$ to $K[\boldsymbol{x}\setminus\boldsymbol{s}]$.*

Given a K-algebra homomorphism

$$\phi : K[\boldsymbol{y}] = K[y_1, \dots, y_m] \longrightarrow K[\boldsymbol{x}]/I\,,\; y_i \longmapsto \overline{f}_i := f_i + I,$$

its kernel can be computed via elimination:

Proposition 2.31 (Kernel of a Ring Map). *Let J be the ideal*

$$J = IK[\boldsymbol{x},\boldsymbol{y}] + \big\langle f_1 - y_1, \dots, f_m - y_m \big\rangle \subset K[\boldsymbol{x},\boldsymbol{y}].$$

Then

$$\ker\phi = J \cap K[\boldsymbol{y}].$$

Computing $\ker\phi$ means to compute the K-algebra relations on $\overline{f}_1, \dots, \overline{f}_m$ and, thus, to represent the subalgebra $K\big[\overline{f}_1, \dots, \overline{f}_m\big]$ of $K[\boldsymbol{x}]/I$ as an affine ring: $K\big[\overline{f}_1, \dots, \overline{f}_m\big] \cong K[\boldsymbol{y}]/\ker\phi$. Geometrically, as already pointed out, computing kernels of ring maps means to compute the **Zariski closure of the image of an algebraic set under a morphism**. Note that in contrast to the case of affine algebraic sets, the image of a projective algebraic set under a morphism is always Zariski closed.

Example 2.32. We use `SINGULAR` to compute defining equations for the twisted cubic curve $C \subset \mathbb{P}^3(\mathbb{R})$ via its parametrization. Algebraically, this amounts to computing the kernel of the ring map

$$\mathbb{Q}[w,x,y,z] \to \mathbb{Q}[s,t], \ \ w \mapsto s^3, \ x \mapsto s^2t, \ y \mapsto st^2, \ z \mapsto t^3 ,$$

and, thus, to eliminate the variables s, t from the ideal

$$\langle w - s^3, x - s^2t, y - st^2, z - t^3 \rangle \subset \mathbb{Q}[s,t,w,x,y,z] .$$

For this, we set up a ring with a block order having the desired elimination property (see Lecture 3 for more on implementing monomial orders):

```
> ring P1P3 = 0, (s,t,w,x,y,z), (dp(2),dp(4));
> ideal J = w-s3, x-s2t, y-st2, z-t3;
> J = groebner(J);
> J;
J[1]=y2-xz
J[2]=xy-wz
J[3]=x2-wy
J[4]=sz-ty
[...]
J[10]=s3-w
```

The first three Gröbner basis elements do not depend on s and t, they define C. To compute these elements, we may alternatively use the built-in command `preimage` which, hiding the elimination step, computes the desired kernel for us (see Lecture 3, Section 3.6.3):

```
> ring P1 = 0, (s,t), dp;
> ideal ZERO;
> ideal PARA = s3, s2t, st2, t3;
> ring P3 = 0, (w,x,y,z), dp;
> ideal IC = preimage(P1,PARA,ZERO);
> print(IC);
y2-xz,
xy-wz,
x2-wy
```

The point $p = (1:0:1:0) \in \mathbb{P}^3(\mathbb{R})$ does not lie on C:

```
> ideal P = w-y, x, z;
> size(reduce(IC,groebner(P),1));  // ideal membership test
2
```

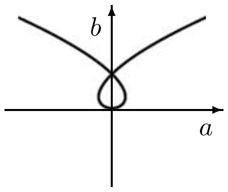
By projecting C from p, we obtain, thus, a morphism $\pi : C \to \mathbb{P}^2(\mathbb{R})$. This morphism is defined by the linear forms $w-y$, x, z defining p. We compute the image $\pi(C)$, that is, the kernel of the ring map

$$\mathbb{Q}[a,b,c] \to \mathbb{Q}[w,x,y,z]/\langle y^2 - xz, xy - wz, x^2 - wy\rangle,$$
$$a \mapsto \overline{w} - \overline{y}, \ \ b \mapsto \overline{x}, \ \ c \mapsto \overline{z} ,$$

where a, b, c are the homogenous coordinates on $\mathbb{P}^2(\mathbb{R})$:

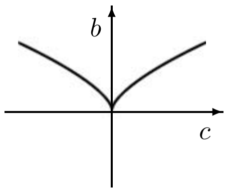
```
> ring P2 = 0, (a,b,c), dp;
> ideal PIC = preimage(P3,P,IC);
> PIC;
PIC[1]=b3-a2c-2b2c+bc2
```

The projected curve is a nodal cubic curve which we visualize in the affine chart $\mathbb{A}^2(\mathbb{R}) \cong \mathbb{P}^2(\mathbb{R}) \setminus V(c)$. For this, we use SURF:



Proceeding similarly for the point $q = (0 : 1 : 0 : 0)$, we get a cuspidal cubic curve (see Lecture 3 for the `setring` command used below):

```
> setring P3;
> ideal Q = w, y, z;
> size(reduce(IC,groebner(Q),1));   // check: Q not on C
1
> setring P2;
> ideal QIC = preimage(P3,Q,IC);
> QIC;
QIC[1]=b3-ac2
```



□

Example 2.33. Consider the map

$$S^2 \longrightarrow \mathbb{A}^3(\mathbb{R})\,, \quad (x_1, x_2, x_3) \longmapsto (x_1x_2, x_1x_3, x_2x_3),$$

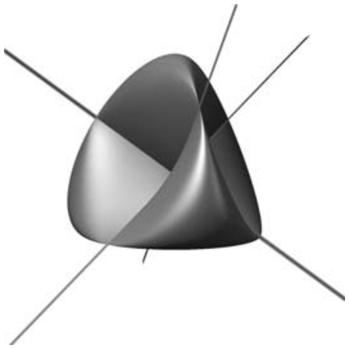
from the real 2-sphere

$$S^2 = V(x_1^2 + x_2^2 + x_3^2 - 1) \subset \mathbb{A}^3(\mathbb{R})$$

to the real 3-space. We refer to (the closure of) its image as the **Steiner Roman surface**. Using the `preimage` command, we compute a defining equation for this surface:

```
> ring S2 = 0, x(1..3), dp;
> ideal SPHERE = x(1)^2+x(2)^2+x(3)^2-1;
> ideal MAP = x(1)*x(2), x(1)*x(3), x(2)*x(3);
> ring R3 = 0, y(1..3), dp;
> ideal ST = preimage(S2, MAP, SPHERE);
> print(ST);
y(1)^2*y(2)^2+y(1)^2*y(3)^2+y(2)^2*y(3)^2-y(1)*y(2)*y(3)
```

To visualize the Steiner Roman surface, we again use SURF:



Note that the Steiner Roman surface is irreducible since S^2 is irreducible. What points in the picture are not in the image of S^2? □

Exercise 2.34. Show that the real algebraic set in Example 2.1 is the closure of the image of the map $S^2 \to \mathbb{A}^3(\mathbb{R})$, $(x_1, x_2, x_3) \mapsto (x_1, x_2, x_2 x_3)$. Conclude that this algebraic set is irreducible. □

Finally, we explain how to compute the homogenization of an ideal with respect to an extra variable (in general, as we will see in Exercise 2.2, it is *not* enough to just homogenize the given generators).

Proposition 2.35. *Let I be an ideal of $K[\boldsymbol{x}] = K[x_1, \dots, x_n]$. Pick a global monomial order $>$ on $K[\boldsymbol{x}]$ which is* **degree compatible**, *that is, which satisfies* $(\deg \boldsymbol{x}^\alpha > \deg \boldsymbol{x}^\beta \implies \boldsymbol{x}^\alpha > \boldsymbol{x}^\beta)$. *If x_0 is an extra variable, set*

$$\boldsymbol{x}^\alpha x_0^d >_{\text{hom}} \boldsymbol{x}^\beta x_0^e \ :\Longleftrightarrow\ \boldsymbol{x}^\alpha > \boldsymbol{x}^\beta \ \ \textit{or}\ \ (\boldsymbol{x}^\alpha = \boldsymbol{x}^\beta \ \textit{and}\ d > e).$$

Then $>_{\text{hom}}$ is a global monomial order on $K[x_0, x_1, \dots, x_n]$ (in fact, it is a product order combining two global monomial orders). Further, the following holds if we homogenize with respect to x_0: if $f_1, \dots, f_r$ form a Gröbner basis for I with respect to $>$, the homogenized polynomials $f_1^{\text{hom}}, \dots, f_r^{\text{hom}}$ form a Gröbner basis for the homogenized ideal I^{hom} with respect to $>_{\text{hom}}$.

Remark 2.36 (Further Reading). For more details and proofs of the results presented in this lecture, see Cox, Little, and O'Shea (1997), Decker and Schreyer (2006), Eisenbud (1995), Greuel and Pfister (2002), and Matsumura (1986).

Lecture 3

An Introduction to SINGULAR

The material presented in this lecture is meant to guide SINGULAR users from taking their first steps into SINGULAR to writing extensive SINGULAR libraries. We treat SINGULAR basics such as the implementation of rings with monomial orders, and we explain more advanced techniques which make Gröbner bases computations with respect to "slow" monomial orders feasible. We consider the computational problems discussed in Lecture 2 from a SINGULAR point of view, and we address the handling of graded modules and the computation of syzygies. We also include a section on computing in noncommutative algebras. Finally, we discuss how to write and debug libraries, how to communicate with other computer algebra systems, and how to access SURF from SINGULAR.

3.1 General Remarks on SINGULAR and its Syntax

SINGULAR is a computer algebra system designed for polynomial computations, with special emphasis on the needs of commutative algebra, algebraic geometry, and singularity theory. It is a free software under GNU Public License, available for various platforms. To obtain SINGULAR, download it from its homepage

`http://www.singular.uni-kl.de`

and install it following the instructions (at this writing, you need to download up to three files, platform depending). The components of SINGULAR include a precompiled C/C++ program, referred to as the SINGULAR **kernel**, several **libraries**, and the **on-line help system**.

In the kernel, the core algorithms for polynomial computations are implemented. These include algorithms for arithmetic operations, for Gröbner basis and syzygy computations, and for polynomial factorization.

The libraries add further computational tools to SINGULAR, thus widely extending the functionality provided by the kernel. Each library is a text file consisting of a collection of procedures written in the SINGULAR **user language**. This language is interpreted, not compiled. It resembles C:

assignments	⟨variable⟩ `=` ⟨value⟩
comparisons	`==`, `<`, `>`, `!=`
conditional statements	`if`, `else`
loops	`for`, `while`
blocks	{ }
comments	`//`, `/* */`

In a SINGULAR session, the user enters commands at the key board in response to the input prompt `>` which is offered by the system. Each command ends with a semicolon. Entering a command makes the system perform computations, display the results on the screen, and offer a new input prompt. For instance:

```
> 2+3+4;
9
>
```

Here, the first line is the input line. One input line may consist of several commands:

```
> 2+3; 3+5;
5
8
```

To cut lengthy input into handy pieces, press the `enter` button at the keyboard without typing a semicolon. Then SINGULAR will offer the prompt `.` to start a new line on the screen and wait for further input. For instance:

```
> 1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1
. +1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1;
42
```

The **on-line help system** provides information on what commands are available and how to use them. Entering

```
> help;
```

makes SINGULAR display the title page of its on-line manual which offers a short table of contents. In response to typing `help` ⟨`topic`⟩, SINGULAR will print the available documentation on ⟨`topic`⟩. Here, ⟨`topic`⟩ may be any index entry of the SINGULAR manual (for instance, the name of a command such as `groebner`). At this writing, a link to the index is provided towards the end of the title page of the on-line manual. The help information is displayed using the default help browser. Platform depending, this may, for instance, be a web browser. Type `help browsers;` for a list of the supported browsers and for information on how to switch to another browser.

Most SINGULAR **libraries** are not immediately accessible and have to be loaded into a SINGULAR session, for instance, by using the LIB command as in Lecture 2, Example 2.27:

```
> LIB "random.lib";  // loads other libraries incl. matrix.lib
>                    // and elim.lib, too
```

At this writing, SINGULAR offers libraries for computations in linear algebra, commutative algebra, singularity theory, invariant theory, symbolic-numerical solving, visualization and coding theory. Type `help Singular libraries;` and follow the respective links to see what is available.

Having so far in this lecture discussed SINGULAR features which are similar to those of general purpose computer algebra systems such as MAPLE, MATHEMATICA, and MUPAD, we now come to an important difference. Namely, *to define and manipulate polynomial data in* SINGULAR, *a ring has to be defined first.* Otherwise, it is not even possible to add numbers other than those of type `int` (integers in a certain range):

```
> 1/3 + 1/5;
   ? no ring active
   ? error occurred in STDIN line ...: `1/3 + 1/5;`
```

At first glance, the user may dislike this feature. However, it fits well with handling mathematical problems coming from algebra and geometry. In algebraic geometry, properties of quotient rings of polynomial rings reflect properties of algebraic sets. To investigate local properties, we study localizations of polynomial rings at maximal ideals and quotient rings thereof. Further, morphisms between algebraic sets correspond to ring homomorphisms on the algebraic side. As a consequence, several distinct rings may have to be considered in a SINGULAR session, along with ring homomorphisms between them. To avoid confusion, we need to be able to name and access rings the same way simpler objects can be named and accessed in other programming languages.

The names of almost all **data types** in SINGULAR are reminiscent of mathematical objects. Most of them are **ring dependent types**:

Ring independent types	`int`, `intmat`, `intvec`, `string`
Ring types	`ring`, `qring`
Ring dependent types	`ideal`, `map`, `matrix`, `module`, `number`, `poly`, `resolution`, `vector`
Special types	`link`, `list`, `proc`, `package`

In the following two sections, we explain in some detail how rings can be implemented in SINGULAR (as objects of types `ring` or `qring`), how to create ring maps between them (of type `map`), and how to represent the basic mathematical objects (ideal, vector, module) with the help of the respective data types (`ideal`, `vector`, `module`).

3.2 Rings in SINGULAR

The basic rings in SINGULAR are polynomial rings with coefficients in a field and certain localizations thereof. They are implemented using the `ring` command. In addition, the `qring` command allows us to define quotient rings of one of the basic rings already implemented.

Each ring in SINGULAR carries a monomial order. To implement a polynomial ring, choose a global monomial order[1] as in the input line below:

```
> ring R = 0, (x,y), dp;
```

The definition consists of a part naming the ring (here, R) and three further parts, declaring

- the *coefficient field*,
- the *variables*, and
- the *monomial order*.

The basic coefficient fields in SINGULAR are prime fields, specified by their *characteristic*. The 0 in the input line above refers to the prime field of characteristic zero, that is, to the field $\mathbb{Q}$ of rational numbers. A positive prime p instead would declare the prime field $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$ with p elements. Entering *parameters* in addition to the characteristic, the coefficient field will be an extension field of the specified prime field.

Example 3.1 (Ring Definitions I: Polynomial Rings). We give examples of rings over various coefficient fields (always choosing the degree reverse lexicographic order).

(1) $\mathbb{Q}[x_1, \ldots, x_7]$:

```
> ring R1 = 0, x(1..7), dp;
```

(2) $\mathbb{Q}(\sqrt{-1})[x, y]$:

```
> ring R2 = (0,i), (x,y), dp;
> minpoly = i^2+1;
```

A single parameter (here, `i`) defines a simple algebraic extension of the prime field if a minimal polynomial is specified (here, `i^2+1`). In this case, the parameter is treated as a primitive element of the extension. See Lecture 6, Remark 6.15 for further information.

(3) $\mathbb{F}_8[x, y]$:

```
> ring R3 = (2,a), (x,y), dp;
> minpoly = a^3+a+1;
```

[1] Localizations of polynomial rings are defined by local and mixed orders; see Lecture 9.

Due to a different internal representation, the arithmetic operations are faster if we implement $\mathbb{F}_8[x,y]$ as follows:

```
> ring R3prime = (2^3,a), (x,y), dp;
```

(4) $\mathbb{F}_{32003}(s,t)[x,y]$:

```
> ring R4 = (32003,s,t), (x,y), dp;
```

Without any further specification, parameters define a purely transcendental extension of the prime field (here, the rational function field $\mathbb{F}_{32003}(s,t)$). $\square$

Next, we give an example of how to define quotient rings of polynomial rings using the `qring` command.

Example 3.2 (Ring Definitions II: Quotient Rings).
(5) The homogeneous coordinate ring of the twisted cubic curve in $\mathbb{P}^3$ may be implemented as follows:

```
> ring R5 = 0, (w,x,y,z), dp;
> matrix m[2][3] = w,x,y,x,y,z;
> ideal I = minor(m,2);
> qring Q = groebner(I);
```
$\square$

We refer to Remark 3.53 for an alternative way of defining objects of type `ring` or `qring` (using the `ringlist` command). Moreover, we refer to Section 3.7 for the definition of `ring`s implementing noncommutative GR-algebras.

Remark 3.3 (Active Ring). At each stage of a SINGULAR session, at most one of the rings already defined is **active** in the following sense:

- Ring dependent variables are directly accessible only if they belong to the active ring.
- Computations involving ring dependent objects are only carried through if the objects belong to the active ring. $\square$

Entering `basering;`, you will get information on the active ring:

```
> basering;
//   characteristic : 0
//   number of vars : 4
//        block   1 : ordering dp
//                  : names    w x y z
//        block   2 : ordering C
// quotient ring from ideal
_[1]=y2-xz
_[2]=xy-wz
_[3]=x2-wy
```

Here, the active ring is R5 (each definition of a new ring makes this new ring the active ring). Among others, the output shows that R5 comes equipped with the order dp. See Example 3.8 for the meaning of `block 2 : ordering C`.

To switch to one of the other rings already defined, use `setring`:

```
> setring R2;
> 1/3+1/5;
8/15
> setring R3;
> 1/3+1/5;
0
```

SINGULAR stores and displays a polynomial such that its terms are sorted with respect to the monomial order on the ring it depends on. The leading term is first, the smallest term is last. In each of the examples above, we have chosen the degree reverse lexicographic order $>_{\mathtt{dp}}$ which depends on how the variables are sorted. In SINGULAR, they are sorted according to their appearance in the ring definition:

```
> varstr(R3);
x,y
> poly f = x+y;
> f;
x+y
> ring S3 = (2^3,a), (y,x), dp;
> poly f = x+y;
> f;
y+x
```

3.2.1 Global Monomial Orders

In describing some global monomial orders in SINGULAR, we suppose that our polynomial ring is $K[\boldsymbol{x}] = K[x_1, \dots, x_n]$. We should point out that for a beginner in SINGULAR, there is no need to study the details of this section. Usually, the following advice is sufficient:

- If no elimination order is required, use the degree reverse lexicographic order $>_{\mathtt{dp}}$.
- If an elimination order is required, use the lexicographic order $>_{\mathtt{lp}}$ only if the elimination property with respect to *each* initial set of variables $x_1, \dots, x_k$, $k = 1, \dots, n-1$, is needed. To eliminate a *specific* initial set of variables $x_1, \dots, x_k$, the product order `(dp(k),dp(n-k))` is usually more effective. See Example 2.29 and Section 3.6.2 for elimination.

On the other hand, some applications of Buchberger's algorithm require the choice of global monomial orders other than $>_{\mathtt{dp}}$, $>_{\mathtt{lp}}$, and product orders combining these. Here is an overview on how global monomial orders are implemented in SINGULAR.

A. Predefined Orders

The following global monomial orders are predefined in SINGULAR:

`lp`	lexicographic order
`Dp`	degree lexicographic order
`Wp(`$\boldsymbol{w}$`)`	weighted degree lexicographic order $\big(\boldsymbol{w} = (w_1, \ldots, w_n) \in \mathbb{Z}^n_{\geq 0}\big)$
`rp`	reverse lexicographic order
`dp`	degree reverse lexicographic order
`wp(`$\boldsymbol{w}$`)`	weighted degree reverse lexicographic order $\big(\boldsymbol{w} = (w_1, \ldots, w_n) \in \mathbb{Z}^n_{>0}\big)$

The lexicographic and the degree reverse lexicographic order are already known to us. Before defining the other orders, we introduce the following notation.

Definition 3.4. If $\boldsymbol{w} = (w_1, \ldots, w_n) \in \mathbb{R}^n$ is any vector, and if $\boldsymbol{x}^\alpha \in K[\boldsymbol{x}]$ is a monomial, we set

$$\boldsymbol{w}\text{-deg}(\boldsymbol{x}^\alpha) = \sum_{i=1}^n w_i \alpha_i .$$

We refer to $\boldsymbol{w}$ as a **weight vector** and define the **$\boldsymbol{w}$-weighted degree** of an arbitrary polynomial $0 \neq f = \sum_\alpha c_\alpha \boldsymbol{x}^\alpha \in K[\boldsymbol{x}]$ to be

$$\boldsymbol{w}\text{-deg}(f) = \max\left\{\boldsymbol{w}\text{-deg}(\boldsymbol{x}^\alpha) \;\middle|\; c_\alpha \neq 0\right\}. \qquad \square$$

We now define the other orders listed above:

$$\begin{aligned}
\boldsymbol{x}^\alpha >_{\texttt{Dp}} \boldsymbol{x}^\beta \;:\Longleftrightarrow\;& \deg(\boldsymbol{x}^\alpha) > \deg(\boldsymbol{x}^\beta) \text{ or} \\
& \big(\deg(\boldsymbol{x}^\alpha) = \deg(\boldsymbol{x}^\beta) \text{ and the first nonzero entry of } \alpha - \beta \text{ is positive}\big). \\
\boldsymbol{x}^\alpha >_{\texttt{Wp}(\boldsymbol{w})} \boldsymbol{x}^\beta \;:\Longleftrightarrow\;& \boldsymbol{w}\text{-deg}(\boldsymbol{x}^\alpha) > \boldsymbol{w}\text{-deg}(\boldsymbol{x}^\beta) \text{ or} \\
& \big(\boldsymbol{w}\text{-deg}(\boldsymbol{x}^\alpha) = \boldsymbol{w}\text{-deg}(\boldsymbol{x}^\beta) \text{ and the first nonzero entry of } \alpha - \beta \text{ is positive}\big). \\
\boldsymbol{x}^\alpha >_{\texttt{rp}} \boldsymbol{x}^\beta \;:\Longleftrightarrow\;& \text{the last nonzero entry of } \alpha - \beta \text{ is negative.} \\
\boldsymbol{x}^\alpha >_{\texttt{wp}(\boldsymbol{w})} \boldsymbol{x}^\beta \;:\Longleftrightarrow\;& \boldsymbol{w}\text{-deg}(\boldsymbol{x}^\alpha) > \boldsymbol{w}\text{-deg}(\boldsymbol{x}^\beta) \text{ or} \\
& \big(\boldsymbol{w}\text{-deg}(\boldsymbol{x}^\alpha) = \boldsymbol{w}\text{-deg}(\boldsymbol{x}^\beta) \text{ and the last nonzero entry of } \alpha - \beta \text{ is negative}\big).
\end{aligned}$$

B. Product Orders

As already remarked, the main use made of product orders is elimination (for another application, see Lecture 5, Example 5.15). In Example 2.29, we introduced the product of two monomial orders. Inductively, we get product orders combining three and more monomial orders. Note that a product order is global iff all its components are global.

In SINGULAR, any combination of predefined monomial orders to a product order can be implemented. For instance, entering

```
> ring R = 0, x(1..7), (dp(3),wp(2,1),dp);
```

defines the polynomial ring $\mathbb{Q}[x_1, \ldots, x_7]$ equipped with the global product order $(>_1, >_2, >_3)$, where

- $>_1$ is the degree reverse lexicographic order on $\mathbb{Q}[x_1, x_2, x_3]$,
- $>_2$ is the (2,1)-weighted degree reverse lexicographic order on $\mathbb{Q}[x_4, x_5]$, and
- $>_3$ is the degree reverse lexicographic order on $\mathbb{Q}[x_6, x_7]$.

Typing `basering;`, we get the information below:

```
//   characteristic : 0
//   number of vars : 7
//        block   1 : ordering dp
//                  : names    x(1) x(2) x(3)
//        block   2 : ordering wp
//                  : names    x(4) x(5)
//                  : weights     2    1
//        block   3 : ordering dp
//                  : names    x(6) x(7)
//        block   4 : ordering C
```

C. Matrix Orders

By a result of Robbiano (1985), each monomial order on $K[\boldsymbol{x}]$ can be realized as a matrix order $>_M$ for some matrix $M \in \mathrm{GL}(n, \mathbb{R})$ (see Greuel and Pfister (2002), Exercise 1.2.9 for some hints on the proof).

Definition 3.5. Let $M \in \mathrm{Mat}(m \times n, \mathbb{R})$ be a matrix of rank n, for some $m \geq n$. The **matrix order** $>_M$ on $K[\boldsymbol{x}]$ is defined by

$$\boldsymbol{x}^\alpha >_M \boldsymbol{x}^\beta \;:\Longleftrightarrow\; M \cdot \alpha >_{\mathtt{lp}} M \cdot \beta.$$

Here, the right-hand side means that the first nonzero entry of $M \cdot (\alpha - \beta)$ is positive. □

In this way, we get a monomial order which is global iff the first nonzero entry of each column of M is positive.

Example 3.6. The input line below implements a ring equipped with a global matrix order. In fact, this order coincides with $>_{\mathtt{dp}}$:

```
> ring R = 0, (x,y,z), M(1,1,1, 0,0,-1, 0,-1,0);
```
□

Note that evaluating a matrix order may be time consuming. Hence, SINGULAR computations with respect to a predefined order (here, dp) are usually faster than computations with the same order implemented as a matrix order.

Remark 3.7 (Internal Limitations). In SINGULAR, only matrix orders defined by a quadratic integer matrix M which is invertible over $\mathbb{Q}$ can be implemented. In fact, the entries of M have to be of type int and are, thus, integers of a rather limited range. The same applies to the entries of weight vectors. For information on the range, type

```
> help limitations;
```

□

D. Orders with an Extra Weight Vector

Given a weight vector $\boldsymbol{w} \in \mathbb{R}^n$ and a monomial order $>$ on $K[\boldsymbol{x}]$, we get a new monomial order $>_{\mathtt{a}(\boldsymbol{w})}$ on $K[\boldsymbol{x}]$ by setting

$$\boldsymbol{x}^\alpha >_{\mathtt{a}(\boldsymbol{w})} \boldsymbol{x}^\beta :\Longleftrightarrow \boldsymbol{w}\text{-deg}(\alpha) > \boldsymbol{w}\text{-deg}(\beta)\,,\ \text{or}$$
$$\big(\boldsymbol{w}\text{-deg}(\alpha) = \boldsymbol{w}\text{-deg}(\beta) \text{ and } \boldsymbol{x}^\alpha > \boldsymbol{x}^\beta\big)\,.$$

In other words, a matrix defining $>_{\mathtt{a}(\boldsymbol{w})}$ is obtained from a matrix defining $>$ by inserting $\boldsymbol{w}$ as the top row. The following holds:

- $>_{\mathtt{a}(\boldsymbol{w})}$ is global if the w_i are strictly positive, or if the w_i are nonnegative and $>$ is global;
- $>_{\mathtt{a}(\boldsymbol{w})}$ has the elimination property with respect to the set of variables $\{x_i \mid w_i > 0\}$.

The line below, for instance, implements $\mathbb{Q}[x,y,z,w]$ equipped with a global monomial order which has the elimination property with respect to $\{x,w\}$:

```
> ring R = 0, (x,y,z,w), (a(1,0,0,1),dp);
```

E. Monomial Orders on Free Modules

Let F be a free $K[\boldsymbol{x}]$-module with a fixed basis $e_1,\dots,e_s$. There are many ways of extending a given monomial order $>$ on $K[\boldsymbol{x}]$ to a monomial order on F. Most notably, we define:

$$\boldsymbol{x}^\alpha e_i >_{(\mathtt{c},>)} \boldsymbol{x}^\beta e_j :\Longleftrightarrow i < j \ \text{ or } \ \big(i = j \text{ and } \boldsymbol{x}^\alpha > \boldsymbol{x}^\beta\big)\,,$$
$$\boldsymbol{x}^\alpha e_i >_{(\mathtt{C},>)} \boldsymbol{x}^\beta e_j :\Longleftrightarrow i > j \ \text{ or } \ \big(i = j \text{ and } \boldsymbol{x}^\alpha > \boldsymbol{x}^\beta\big)\,,$$

giving *priority to the basis vectors* $e_1,\dots,e_s \in F$, and

$$\boldsymbol{x}^\alpha e_i >_{(>,\mathtt{c})} \boldsymbol{x}^\beta e_j :\Longleftrightarrow \boldsymbol{x}^\alpha > \boldsymbol{x}^\beta \ \text{ or } \ \big(\boldsymbol{x}^\alpha = \boldsymbol{x}^\beta \text{ and } i < j\big)\,,$$
$$\boldsymbol{x}^\alpha e_i >_{(>,\mathtt{C})} \boldsymbol{x}^\beta e_j :\Longleftrightarrow \boldsymbol{x}^\alpha > \boldsymbol{x}^\beta \ \text{ or } \ \big(\boldsymbol{x}^\alpha = \boldsymbol{x}^\beta \text{ and } i > j\big)\,,$$

giving *priority to the monomials in* $K[\boldsymbol{x}]$. By definition, an order extending $>$ is global iff $>$ is global.

Example 3.8. In SINGULAR, the type of extension has to be chosen when defining a ring. This choice will then be applied to all free modules over that ring. In all examples considered so far, we only specified a monomial order $>$ on the ring itself. In this case, SINGULAR will select the default extension $>_{(>,\mathtt{C})}$. This explains the output displayed in response to the `basering` command on Page 67:

```
//        block   1 : ordering dp
//                  : names    w x y z
//        block   2 : ordering C
```

To define rings with other extensions, type, for instance,

```
> ring R = 0, (x,y), (c,dp);
```

Entering `basering;`, the output now includes the lines below:

```
//        block   1 : ordering c
//        block   2 : ordering dp
//                  : names    x y
```

$\square$

Remark 3.9. For some applications, other ways of extending a monomial order $>$ on $K[\boldsymbol{x}]$ to one on F are needed. For instance, in addition to giving weights to the variables x_k, it may be necessary to give weights to the vectors e_i. Implementing a monomial order on F which first compares the resulting weighted degrees of the monomials $\boldsymbol{x}^\alpha e_i$, using $>$ as a tie breaker if needed, is more subtle. To do this, define a ring with extra variables `e(1)`, …, `e(s)`, choosing the monomial order appropriately, and compute in the quotient ring modulo the ideal spanned by the products `e(i)e(j)`, $i, j = 1, \dots, s$. Here is an example of how to set this up (assigning the weights $-1, -2, -4$ to the three basis vectors):

```
> ring R = 0, (x,y,e(1..3)), wp(1,1,-1,-2,-4);
> ideal I = e(1..3);
> qring Q = groebner(I^2);
```

$\square$

3.2.2 Creating Ring Maps

As already remarked, ring dependent objects are directly accessible only if they belong to the active ring. For instance:

```
> ring R = 0, (x,y), dp;
> int i = 1;              // object of ring independent type int
> poly f = x;             // object of ring dependent type poly
> ring S = 0, (x,y), dp;  // active ring is changed
> poly g = y;
> f;
   ? `f` is undefined
   ? error occurred in STDIN line 6: `f;`
```

Entering `listvar();` makes SINGULAR print all objects directly accessible from the active ring (which is marked by an asterisque):

```
> listvar();
// S                      [0]  *ring
//      g                      [0]  poly
// i                      [0]  int 1
// R                      [0]  ring
```

We see that SINGULAR lists the ring objects, the objects depending on the active ring, and the ring independent objects.

In many cases, one should like to transfer objects (here, `f`) from one ring to another. This is done via ring homomorphisms (ring maps, for short). In SINGULAR, the target of a ring map is always the active ring:

$$\text{ring map}: \text{ preimage ring} \longrightarrow \text{active ring}.$$

In describing the implementation of ring maps, we suppose that m is the number of variables of the preimage ring.

We can use the data type `map` to set up arbitrary ring maps. An object of type `map` is specified by giving the name of the preimage ring and a list of polynomials $f_1, \ldots, f_m$ of the active ring. Mathematically, we get the unique ring map sending the ith variable of the preimage ring to f_i, for all i.

The commands `fetch` and `imap` implement special ring maps. They are particularly convenient for dealing with inclusions of rings and for changing the monomial order of the active ring. Using `fetch`, the ith variable of the preimage ring is sent to the ith variable of the active ring, or to zero if i exceeds the number of variables of the active ring. Using `imap`, a variable of the preimage ring is sent to the variable with the same name, if such a variable exists in the active ring, and to zero, otherwise.

Example 3.10 (Definition of Ring Maps).
(1) User-defined `maps`:

```
> ring R = 0, (x,y), dp;
> poly f = x2+y;
> ring S = 0, (a,b,c), dp;
> map F = R, a-b, c;  // map F: R->S, sending x to a-b, y to c
> poly g = F(f);      // apply the map
> g;
a2-2ab+b2+c
> ring S1 = 2, (a,b,c), lp;
> qring Q = std(a^2);
> map F1 = R, a-b, c; // target ring is qring, with another
>                     // characteristic and monomial order
. poly g=F1(f);
> g;                  // polynomial is not yet reduced
a2+b2+c
> reduce(g,std(0));
b2+c
```

(2) Special maps using `imap` and `fetch`:

```
> ring R1 = 0, (a,b,c,x,y,z), dp;
> fetch(R,f);          // fetch preserves order of variables
a2+b
> imap(R,f);           // imap preserves names of variables
x2+y
> fetch(Q,g);
a2+b2+c
```
□

3.3 Ideals, Vectors and Modules in SINGULAR

An **ideal** is specified by a comma separated list of generating polynomials. For instance:

```
> ring R = 0, (x,y,z), dp;
> ideal I = x2-y, y4-z2;
```

Alternatively:

```
> ring R = 0, (x,y,z), dp;
> poly f = x2-y;
> poly g = y4-z2;
> ideal I = f,g;
```

Note that SINGULAR understands input in a short (for instance, `x3-2y2`) and a long (for instance, `x^3-2*y^2`) format. Whether the output is displayed in the short or the long format depends on the active ring (type `help short;` for information). To enforce the long format, enter

```
> short = 0;
```

This is particularly useful when communicating with other computer algebra systems (see Section 3.9).

A **vector**, that is, an element of a free module, is given as a list of polynomials either in a dense format,

```
> vector v = [f,0,0,g,0];
```

or in a sparse format,

```
> vector v = f*gen(1)+g*gen(4);
```

Independent of the input format, SINGULAR stores a vector in a sparse representation. As a consequence, objects of type `vector` do not carry information on the rank of the free module they are supposed to belong to. In particular, the addition of vectors of different length is allowed (see the SINGULAR code on the following page).

The output format for a vector depends on the chosen monomial order for free modules over the active ring. If the order is of type `(c,>)`, the dense format is used:

```
> ring R = 0, (x,y,z), (c,dp);
> vector v = [x,y]+[x2,1,z3,0];
> v;
[x2+x,y+1,z3]
```

Otherwise, SINGULAR displays vectors in the sparse format:

```
> ring S = 0, (x,y,z), (dp,c);
> vector v = fetch(R,v);
> v;
z3*gen(3)+x2*gen(1)+x*gen(1)+y*gen(2)+gen(2)
```

In any case, the dense format can be enforced by using the print command:

```
> print(v);
[x2+x,y+1,z3]
```

Implementing **modules** in SINGULAR is more subtle, and, in fact, somewhat confusing. The corresponding data types are module and matrix. A SINGULAR object of type module is implemented as a comma separated list of objects of type vector which mathematically are thought of as column vectors. For instance:

```
> ring R = 0, (x,y), dp;
> module I = [x2,-y,y,-y,0], [0,0,y], [y,x];
> print(I);
x2,0,y,
-y,0,x,
y, y,0,
-y,0,0
```

Given a module I in SINGULAR, we usually think of it as the submodule I of a free module F (in our example above, $F = \mathbb{Q}[x,y]^4$). Depending on the context, however, we may sometimes have to interprete I as a different mathematical object, namely the quotient module $M = F/I$. That is, the polynomial vectors generating I in SINGULAR are thought of as the columns of a presentation matrix of M. The latter applies in particular to most of the commands provided by the library homolog.lib. We give two examples of what may happen in the context of homological algebra:

- The command modulo from the SINGULAR kernel refers to the interpretation as submodules, while
- the command Hom from homolog.lib refers to the interpretation as quotient modules

(see Lecture 4 for a detailed discussion of these commands).

An object of type matrix is specified by the number of its rows and columns together with a comma separated list of its entries:

```
> matrix MI[4][3] =  x2, 0, y,
.                   -y, 0, x,
.                    y, y, 0,
.                   -y, 0, 0;
```

In our SINGULAR session, having already implemented the `module I`, we could have defined the `matrix MI` also via **type conversion** from `module` to `matrix`:

```
> matrix MI = I;
```

Similarly, there is a type conversion from `matrix` to `module`.

Note that objects of type `module` are stored in a sparse representation, while objects of type `matrix` are stored in a dense representation. Therefore, it is recommended to use the type `module` instead of `matrix` for large input.

3.4 Handling Graded Modules

To explain how SINGULAR deals with graded modules, we start a new SINGULAR session, defining a ring `R` and a module `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];
> print(I);
xz,-yz2,y2z,y3, -z3,-yz2,0,
0, y2,  0,  0,  yz, y2,  0,
-w,0,   -z2,-yz,0,  0,   -wy2+xz2,
-1,-w,  0,  -x, 0,  -w,  -y2,
0, 0,   -x, 0,  -w, 0,   x2
```

A check on degrees shows that the polynomial vectors generating `I` form a graded matrix. More precisely, we may think of `I` as a graded submodule of the graded free R-module

$$F = R \oplus R(-1)^2 \oplus R(-2)^2$$

(or of any twist of this module). We refer to this fact by saying that $(0, 1, 1, 2, 2)$ is an **admissible degree vector** for `I`.

The degree check can be done in SINGULAR. Typing `homog(I)`, SINGULAR will print `1` if the vectors generating `I` form a graded matrix. In this case, SINGULAR will assign an admissible degree vector to `I` as an attribute. We may use the `attrib` command to check that originally, `I` does not come with such an attribute:

```
> attrib(I,"isHomog");     // no attribute => empty output

> homog(I);
1
```

```
> attrib(I,"isHomog");
0,1,1,2,2
```

If an admissible degree vector is assigned, we may read the degrees from a Betti diagram (see Remarks 1.20 and 3.34 for Betti diagrams):

```
> print(betti(I,0),"betti");
           0     1
------------------
    0:     1     -
    1:     2     1
    2:     2     5
    3:     -     1
------------------
total:     5     7
```

To switch to another admissible degree vector, use the `attrib` command as follows:

```
> intvec DV = 2,3,3,4,4;
> attrib(I,"isHomog",DV);
```

Now, we get the numerical information printed below:

```
> attrib(I,"isHomog");
2,3,3,4,4
> print(betti(I,0),"betti");
           0     1
------------------
    2:     1     -
    3:     2     1
    4:     2     5
    5:     -     1
------------------
total:     5     7
```

To access the top degree $d = 2$ in the first column of the Betti diagram and store it for later use, assign the matrix formed by the numbers of the free generators in the Betti diagram to a matrix of type `intmat` (a matrix with entries of type `int`) and make use of the `"rowShift"` attribute assigned to this matrix:

```
> intmat BI = betti(I,0);
> int d = attrib(BI,"rowShift");
> d;
2
```

Remark 3.11. At this writing, `SINGULAR` does not check whether the assigned degree vector really is admissible. Moreover, the assigned attributes are lost under type conversion (for instance, from `module` to `matrix`, or from `resolution` to `list`). □

3.5 Computing Gröbner Bases

The basic version of Buchberger's algorithm as described in Lecture 1 leaves a lot of freedom in carrying out the computational process. The performance of the algorithm depends in a crucial way on strategies for selecting the next S-polynomial in Buchberger's test, and for computing a standard expression of the S-polynomial in terms of the generators obtained that far (some choices made in the division process are more efficient than others). Further, considerable improvements are obtained by implementing criteria for reducing the number of S-polynomials to be actually considered. The two criteria below have proven to be extremely useful:

- The **product criterion** (not extendable to free modules). If $f, g \in K[\boldsymbol{x}]$ are polynomials such that $\mathrm{L}(f)$ and $\mathrm{L}(g)$ have no variables in common, then $\mathrm{S}(f,g)$ has a standard expression in terms of f, g with remainder zero (indeed, to obtain such an expression, rewrite the relation $gf - fg = 0$). Thus, in Buchberger's test, such an S-polynomial need not be considered.
- The **chain criterion**. If f, g, h are generators in Buchberger's test such that $\mathrm{L}(h)$ divides the least common multiple of $\mathrm{L}(f)$ and $\mathrm{L}(g)$, and if the S-polynomials $\mathrm{S}(f,h)$ and $\mathrm{S}(g,h)$ have been dealt with already, then $\mathrm{S}(f,g)$ need not be considered.

We refer to Cox, Little, and O'Shea (1997), Chapter 2, §9 for a discussion of this topic and for further references. The discussion includes a few remarks on selection strategies such as the **normal strategy** and the **sugar strategy** which together with the criteria above play a prominent role in the SINGULAR implementation of Buchberger's algorithm. This implementation is called by the `std` command[2] (see Remark 3.19 later in this section for the `groebner` command). Example 3.20 will show the implementation at work, illustrating, in particular, the usefulness of the criteria.

Remark 3.12. (1) The arithmetic operations over $\mathbb{Q}$ are much more expensive than those over $\mathbb{Z}$. Therefore, given polynomials with rational coefficients, SINGULAR clears denominators before applying Buchberger's algorithm. Starting from a set $\mathcal{F}$ of polynomials with integer coefficients[3], all computations in Buchberger's test take place over $\mathbb{Z}$. Thus, if $\mathcal{G}$ is the resulting set of polynomials, the elements of $\mathcal{G}$ will have integer coefficients as well. These elements form a Gröbner basis for the ideal generated by $\mathcal{F}$ over $\mathbb{Q}$. The ideal generated by $\mathcal{F}$ over $\mathbb{Z}$, however, may be strictly contained in the ideal generated by $\mathcal{G}$ over $\mathbb{Z}$. This is due to the fact that in the SINGULAR implementation of Buchberger's test, each S-polynomial is divided by the greatest common divisor of its integer coefficients in order to keep the integer coefficients as small as possible. For elements of free modules, SINGULAR proceeds accordingly.

[2] The name `std` refers to **st**an**d**ard bases, see Lecture 9.

[3] In SINGULAR, the elements of a coefficient field are of type `number`. In contrast to integers of type `int`, there is virtually no limitation for the size of integers of type `number`.

(2) A Gröbner basis $\mathcal{G}$ computed with `SINGULAR` is **minimal** in the sense that the leading monomials of the elements of $\mathcal{G}$ form the uniquely determined minimal set of monomial generators for $\mathrm{L}(\langle\mathcal{G}\rangle)$. Depending on the type of the active ring and on the options set, however, $\mathcal{G}$ is not necessarily reduced. Type

```
> help option;
```

for more details. Entering

```
> option(redSB);
```

in a `SINGULAR` session will force `SINGULAR` to compute reduced Gröbner bases. Over $\mathbb{Q}$, however, the leading coefficients will not necessarily be adjusted to 1. To achieve this, use the `simplify` command as in the session below:

```
> ring R = 0, (x,y,z), lp;
> ideal  I = 2y+z,3x-y;
> std(I);
_[1]=2y+z
_[2]=3x-y
> option(redSB);
> ideal G = std(I);
> G;
G[1]=2y+z
G[2]=6x+z
> G = simplify(G,1);
> G;
G[1]=y+1/2z
G[2]=x+1/6z
```

□

As already pointed out, the performance of Buchberger's algorithm is sensitive to the choice of monomial order. A Gröbner basis computation with respect to a less favorable order such as $>_{\mathtt{lp}}$ may easily run out of time or memory even in cases where a Gröbner basis computation with respect to a more efficient order such as $>_{\mathtt{dp}}$ is very well feasible. **Gröbner basis conversion algorithms** and the **Hilbert driven Buchberger algorithm** take their cue from this observation.

A Gröbner basis conversion algorithm proceeds along the following lines:

- Given an ideal $I \subset K[\boldsymbol{x}]$ and a slow monomial order, compute a Gröbner basis for I with respect to an appropriately chosen fast order.
- Convert the result to a Gröbner basis with respect to the given slow order.

Applying the Hilbert driven Buchberger algorithm usually means to proceed in two steps as well:

- Given a homogeneous ideal $I \subset K[\boldsymbol{x}]$ and a slow order, compute a Gröbner basis for I with respect to an appropriately chosen fast order. From the result, compute the Hilbert function of $K[\boldsymbol{x}]/I$.

- Compute a Gröbner basis for I with respect to the given slow order. On your way, make use of the Hilbert function to further reduce the number of S-polynomials which actually need to be considered in Buchberger's test.

Here, the first step is of course superfluous if the Hilbert function of $K[\boldsymbol{x}]/I$ is already known.

At this writing, there are SINGULAR implementations of the conversion algorithm of Faugère, Gianni, Lazard, and Mora (1993) (FGLM for short), of several variants of the Gröbner walk conversion algorithm, and of the Hilbert driven Buchberger algorithm. How to access these implementations will be discussed in what follows. Note that the FGLM algorithm works for zero-dimensional ideals only, while the walk algorithms and the Hilbert-driven approach can be applied to ideals of any dimension.

The FGLM Algorithm

If $I \subset K[\boldsymbol{x}]$ is a zero-dimensional ideal, then $K[\boldsymbol{x}]/I$ is a finite dimensional K-vector space (see Lecture 6, Theorem 6.1). Based on this fact, the FGLM algorithm converts Gröbner bases by means of linear algebra (Gaussian elimination). Its SINGULAR implementation is called by the `fglm` command.

Example 3.13. Using `fglm`, we compute the reduced lexicographic Gröbner basis for the ideal $I \subset \mathbb{Q}[x,y,z]$ generated by the polynomials

$$\begin{aligned} f_1 &= 3x^3y + x^3 + xy^3 + y^2z^2, \\ f_2 &= 2x^3z - xy - xz^3 - y^4 - z^2, \\ f_3 &= 2x^2yz - 2xy^2 + xz^2 - y^4 \end{aligned}$$

(see Amrhein, Gloor, and Küchlin (1997), Example Ex3). To begin with, we compute the reduced Gröbner basis for I with respect to $>_{\mathrm{dp}}$ (initializing an integer by `timer` allows us to print the CPU time used by SINGULAR).

```
> ring R = 0, (x,y,z), dp;
> ideal I = 3x3y+x3+xy3+y2z2, 2x3z-xy-xz3-y4-z2, 2x2yz-2xy2+xz2-y4;
> option(redSB);          // force computation of reduced GBs
> int aa = timer;
> ideal SI = std(I);
> timer-aa;               // time in seconds
0
```

Now, we apply `fglm` to `SI`. For this, `SI` has to be a *reduced* Gröbner basis for a *zero-dimensional* ideal (SINGULAR checks this). The Gröbner basis returned by `fglm` is reduced, too.

```
> ring S = 0, (x,y,z), lp;
> aa = timer;
> ideal J = fglm(R,SI);
> timer-aa;
1
```

We will not display the computed Gröbner basis, as this would fill more than ten pages of our notes. Instead, we use the `size` command to get some information on the Gröbner basis. If applied to an `ideal`, `size` returns the number of nonzero generators. If applied to a `string`, it returns the number of characters. Here are some data describing the reduced lexicographic Gröbner basis:

```
> size(J);                   // number of generators
8
> size(string(J))/68;        // number of lines with 68 characters
>                            // needed to display J:
631
. deg(J[1..size(J)]);        // degrees of the generators
35 34 34 34 34 34 34 34
> leadmonom(J[1..size(J)]); // generators for L(I) w.r.t. lp
z35 yz6 y2z4 y3z2 y5 xz2 xy x3
> leadcoef(J[8]);            // leading coefficient of 8th generator
6440093631623773985969509841859276602512807348986590906348582267651 8
0694267744388309310964144162736424959843858259686293831496555654853 3
87059732896226082504084733570575781959910 4
```

Note that on our machine, the attempt to compute the lexicographic Gröbner basis directly in the ring `S` failed after several hours of running time:

```
> ideal I = fetch(R,I);
> I = std(I);

error: no more memory
```
□

Gröbner Walk Conversion Algorithms

Let $I \subset K[\boldsymbol{x}]$ be an arbitrary ideal. The basic idea of a Gröbner walk algorithm is to approach the target Gröbner basis for I in several steps, "walking" along a path through the Gröbner fan of I (see Sturmfels (1996) for the Gröbner fan). In each step, a Gröbner basis computation with respect to an "intermediate" monomial order is performed. There are several strategies for choosing the path through the Gröbner fan, leading to different variants of the algorithm. See Collart, Kalkbrener and Mall (1997), Amrhein and Gloor (1998), and Tran (2000) for details.

At this writing, the `SINGULAR` implementation of the Gröbner Walk algorithms is affected in its efficiency by the internal limitations on weight vectors described in Remark 3.7 (while walking, each intermediate monomial order is defined as an order with an extra weight vector). Nevertheless, the commands provided by the library `grwalk.lib` (written by I Made Sulandra) often yield a result in cases where a direct Gröbner basis computation fails.

Example 3.14. We recompute the reduced lexicographic Gröbner basis for the ideal I in Example 3.13 by applying the command `fwalk` from `grwalk.lib`:

```
> LIB "grwalk.lib";
> ring S = 0, (x,y,z), lp;
> ideal I = 3x3y+x3+xy3+y2z2, 2x3z-xy-xz3-y4-z2, 2x2yz-2xy2+xz2-y4;
> option(redSB);          // force computation of reduced GBs
> int aa = timer;
> ideal J = fwalk(I);
> timer-aa;
0
```

The `fwalk` command calls a variant of the fractal walk algorithm. Another implementation of this algorithm is provided by the `frwalk` command (which is not part of `grwalk.lib`, but calls a procedure of the `SINGULAR` kernel). □

The Hilbert Driven Buchberger Algorithm

If $I \subset K[\boldsymbol{x}]$ is a homogeneous ideal, and if $>$ is a global monomial order on $K[\boldsymbol{x}]$, the Hilbert function of $K[\boldsymbol{x}]/I$ coincides with that of $K[\boldsymbol{x}]/\mathrm{L}_>(I)$ by Macaulay's Theorem 1.35. In particular, if $\mathcal{G}$ is any set of generators for I, then

$$\dim_K(K[\boldsymbol{x}]/I)_d \leq \dim_K\big(K[\boldsymbol{x}]/\langle \mathrm{L}_>(g) \mid g \in \mathcal{G}\rangle\big)_d$$

for all degrees d. This fact can be used to expedite the computation of a Gröbner basis for I if the Hilbert function of $K[\boldsymbol{x}]/I$ is already known (see Traverso (1996)). Supposing that I is given by homogeneous generators, the resulting Hilbert driven Buchberger algorithm proceeds degree by degree, following in each degree d the basic strategy described below:

(1) The **Hilbert function criterion**. In Buchberger's test, let $\mathcal{G}$ be the set of generators computed that far. If

$$\dim_K(K[\boldsymbol{x}]/I)_d = \dim_K\big(K[\boldsymbol{x}]/\langle \mathrm{L}_>(g) \mid g \in \mathcal{G}\rangle\big)_d,$$

the remaining S-polynomials of degree d need not be considered. In this case, replace d by $d+1$.

(2) If the equality in (1) does not hold, select an S-polynomial $\mathrm{S}(f,g)$ of degree d which has not yet been considered, and compute a remainder of $\mathrm{S}(f,g)$ on division by the elements of $\mathcal{G}$. If the remainder is nonzero, add it to $\mathcal{G}$ and go to (1). If all S-polynomials of degree d have been considered, replace d by $d+1$.

The Hilbert driven Buchberger algorithm is implemented in the `SINGULAR` kernel. It is accessible by calling the `std` command with two input parameters, one of type `ideal`, and one of type `intvec`. Entering a homogeneous ideal $I \subset K[\boldsymbol{x}]$, the integer vector is meant to specify the numerator of the Hilbert series of $K[\boldsymbol{x}]/I$, represented as in Lecture 1, Theorem 1.22.

Remark 3.15. If I is not homogeneous, a Gröbner basis for I may be obtained as follows. Homogenize the given generators for I with respect to a new variable, say, x_0. Extend the given slow order on $K[x_1, \ldots, x_n]$ to an order on $K[x_0, x_1, \ldots, x_n]$ as in Lecture 2, Proposition 2.35. Apply the Hilbert driven Buchberger algorithm to the homogenized generators (first computing a Gröbner basis with respect to a fast order and, thus, the desired Hilbert function, then computing a Gröbner basis with respect to the extended order, making use of the Hilbert function). Dehomogenize the elements of the resulting Gröbner basis. □

Example 3.16. Using the Hilbert driven Buchberger algorithm (homogenizing and dehomogenizing as explained in Remark 3.15), we again recompute the lexicographic Gröbner basis from Example 3.13:

```
> ring S = 0, (x,y,z), lp;
> ideal I = 3x3y+x3+xy3+y2z2, 2x3z-xy-xz3-y4-z2, 2x2yz-2xy2+xz2-y4;
> option(redSB);
> ring Rhom = 0, (x,y,z,t), dp;
> ideal I = imap(S,I);
> ideal Ih = homog(I,t);   // generators of I are homogenized
> int aa = timer;
> Ih = std(Ih);
> timer-aa;
0
> aa = timer;
> intvec H = hilb(Ih,1); timer-aa;
0
> ring Shom = 0, (x,y,z,t), lp;
> ideal Ih = imap(Rhom,Ih);
> aa = timer;
> Ih = std(Ih,H); timer-aa;
0
> Ih = subst(Ih,t,1);
> setring S;
> ideal J = imap(Shom,Ih);
> size(J);
102
```

The result `J` is a lexicographic Gröbner basis, but it is far from being the reduced lexicographic Gröbner basis. To turn `J` into the reduced Gröbner basis (up to adjusting the leading coefficients), we apply the `interred` command:

```
> aa = timer;
> J = interred(J);
> timer-aa;
3
> size(J);
8
```
□

Remark 3.17. Let polynomials $f_1, \ldots, f_r \in \mathbb{Z}[\boldsymbol{x}]$ be given, and let I be the ideal of $\mathbb{Q}[\boldsymbol{x}]$ generated by these polynomials. As we saw in Example 3.13, the coefficients of the polynomials produced by applying Buchberger's algorithm to $f_1, \ldots, f_r$ may be huge. Thus, to compute a Gröbner basis for I using the Hilbert driven Buchberger algorithm, it is often more efficient to perform the Gröbner basis computation yielding the Hilbert function by reducing the given polynomials modulo a sufficiently large prime p (for instance, $p = 32003$). If I_p denotes the ideal of $\mathbb{F}_p[\boldsymbol{x}]$ generated by the images of $f_1, \ldots, f_r$, the Hilbert functions of $\mathbb{Q}[\boldsymbol{x}]/I$ and of $\mathbb{F}_p[\boldsymbol{x}]/I_p$ typically coincide. In examples such as $I = \langle x - 32003y, x\rangle \subset \mathbb{Q}[x, y]$, however, this approach yields a function Φ which differs from the Hilbert function in characteristic zero. Using Φ as a second input parameter for `std`, we may get a result which is not even a generating set for the ideal under consideration. □

Remark 3.18 (The `slimgb` Command). Besides the commands discussed so far, SINGULAR also provides the `slimgb` command for computing Gröbner bases. The underlying algorithm is a variant of Buchberger's algorithm which is specifically designed to reduce the size of the intermediate polynomials produced on its way. It is based on ideas of Faugère and has been developed and implemented in SINGULAR by Brickenstein (2004). As several benchmark examples show, the use of `slimgb` instead of `std` is advisable for Gröbner basis computations over transcendental extensions of prime fields. For instance:

```
> ring R = (32003,a,b,c,d), (t,u,v,w,x,y,z), dp;
> ideal I = -cw+bx, ct+2au-2bu-2cv-(ad+bd),
.           -2tx+4wy+4xz+ct-2aw-2dw-2by-2cz+(ab+bd),
.           t*(z-x)+(a-b+d)*(y-w)+c*(x-z), -tw+a*(t-x)+dx,
.           -2tv+ct-2du+(ad+bd), ct2-(b2-ab+c2)*t-(acd-cd2);
> int aa = timer;
> ideal SI = slimgb(I);
> timer-aa;             // timing for slimgb command
0
> aa = timer;
> SI = std(I);
> timer-aa;             // timing for std command
649
```

This example is taken from the list of benchmark examples provided by the Symbolic Data Project (see `http://www.SymbolicData.org`). In this list, it is referred to as `Geometry/Chou 274 2`. □

Remark 3.19 (The `groebner` Command). Depending on the active ring and some heuristics, the SINGULAR command `groebner` refers to one of the algorithms for computing Gröbner bases discussed in this section. At this writing, it directly calls `std` if the active ring comes equipped with a fast order such as $>_{\mathtt{dp}}$; if the order is $>_{\mathtt{lp}}$, it refers to the Hilbert driven Buchberger algorithm (first computing a Gröbner basis with respect to a fast order and, thus, the Hilbert function, then calling `std` with the Hilbert function as a second input parameter, homogenizing and dehomogenizing, if needed). The `groebner` command is provided by the `SINGULAR` library `standard.lib` which is automatically loaded when starting a `SINGULAR` session. For information on the heuristics behind `groebner`, see the library file `standard.lib`. How to find and read `SINGULAR` library files will be explained on Page 112. □

Example 3.20. We once more compute the lexicographic Gröbner basis from Example 3.13, now using the `groebner` command. Entering `option(prot);` makes `SINGULAR` display some information on how the computation proceeds. In particular, each S-polynomial which needs not be considered due to the Hilbert function criterion is indicated by printing the symbol `h`.

```
> ring S = 0, (x,y,z), lp;
> ideal I = 3x3y+x3+xy3+y2z2, 2x3z-xy-xz3-y4-z2, 2x2yz-2xy2+xz2-y4;
> option(redSB);
> option(prot);
> int aa = timer;
> ideal J = groebner(I);
std in (0),(x,y,z,@t),(dp,C)
[255:1]4(2)sss5s6s7(3)s(4)s(5)s(6)s8(8)s(9)-ss(11)s(12)---9-s(9)-s(
10)--s--10-s(8)s(9)-s---11------
product criterion:9 chain criterion:124
std with hilb in  (0),(x,y,z,@t),(lp(3),C)
[255:1]4(2)sss5ss6s(3)s(5)s7(6)s(7)s(9)s(11)s(13)-s(14)s8(16)s(17)s
(19)s(21)s(23)s(25)s(27)s(28)-s(29)--shhhhh9(24)s(26)s(28)s(30)s(32
)s(33)s(35)s(37)s(39)s(41)shhhhhhhhhhhhhhh10(28)ss(29)s(30)s(32)s(3
4)s(35)s(37)s(39)s(41)s(43)shhhhhhhhhhhhhhhhhh11(26)s(28)s(30)s(32)
s(34)s(35)shhhhhhhhhhhhhhhhhhhhh12(16)s(18)s(20)s(22)s(24)shhhhhhhh
hhhh13(14)s(15)s(17)s(19)s(21)shhhhhhhhhh14(13)s(15)s(17)s(19)shhhh
hhhhhh15(10)s(12)s(14)shhhhhhhh16(8)s(10)s(12)shhhhhh17(8)s(10)s(12
)shhhhhh18(8)s(9)s(11)shhhhhh19(7)s(9)shhhhhh20(5)s(7)shhhh21(5)s(7
)shhhh22(5)s(7)shhhh23(5)s(7)shhhh24(5)s(6)shhhh25(4)shhhh26(2)shh2
7shh28shh29shh30shh31shh32shh33shh34shh35shh36shh37shh38shhhh
product criterion:27 chain criterion:4846
hilbert series criterion:175
dehomogenization
imap to original ring
simplification
interreduction
> timer-aa;
0
> size(J);
8
```

For an interpretation of the various symbols printed above, enter

```
> help option;
```

At this writing, you will find the relevant information at the end of the help text displayed. □

Limitations of Gröbner basis methods

Starting from a few polynomials of low degree and with small coefficients, Buchberger's algorithm may well produce an enormous number of polynomials of high degree and with huge coefficients (as in Examples 3.13 and 3.16). This may happen even though the original set of generators *and* the resulting Gröbner basis are of modest size. That is, the computation may run into the dreaded phenomenon of **intermediate expression swell**. To illustrate this fact, we recompute a simple example given by Adams and Loustaunau (1994), Section 3.3:

Example 3.21. Consider $I = \langle 4x^2y^2+3x,\ y^3+2xy,\ 7x^3+6y\rangle \subset \mathbb{Q}[x,y]$.

```
> ring R = 0, (x,y), dp;
> ideal I = 4x2y2+3x, y3+2xy, 7x3+6y;
> std(I);
_[1]=y
_[2]=x
```

Thus, the reduced Gröbner basis for I is $\{x, y\}$. The coefficients during the computation, however, grow as large as 10^8. In our SINGULAR session, this can be seen by expressing x as a $\mathbb{Q}[x,y]$-linear combination of the original generators. For this, we use the `lift` command which is based on the solution to the Submodule Membership Problem 2.16 (we will discuss the `lift` command in some detail in Sections 3.6.1 and 4.1):

```
> ideal J = x;
> matrix A = lift(I,J);
> A;
A[1,1]=-3670016/18809541x2y+9604/6269847xy2-134217728/131666787y3
       -128/63xy-100352/6269847y2-458752/6269847y+1/3
A[2,1]=536870912/131666787x2y2+401408/6269847x2y+1835008/6269847x2
       -4194304/6269847xy+10976/2089949y2+64/21x+50176/2089949y+
       229376/2089949
A[3,1]=2097152/18809541xy3-5488/6269847y4-25088/6269847y3
       -114688/6269847y2
> matrix(I)*A;
_[1,1]=x
```
□

There is a "worst case" upper bound for the degree of the elements of the reduced Gröbner basis for an ideal $\langle f_1, \ldots, f_r\rangle \subset K[x_1, \ldots, x_n]$ due to Möller and Mora (1984). If d is the maximum degree of the f_i, this bound is

$$2\left(\frac{d^2}{2}+d\right)^{2^{n-1}}.$$

It is, thus, doubly exponential in the number of variables. Examples of Mayr and Meyer (1982) show that the double exponential form of the bound cannot be improved. Despite this high worst case complexity, Buchberger's algorithm works well for many problems of practical interest. The range in the number of variables and degrees in which Gröbner basis computations can be completed, however, is relatively small. See Decker and Schreyer (2001), Section 10 for some benchmarks.

3.6 Basic Applications of Gröbner Bases (revisited)

In this section, we come back to some of the basic problems discussed in Lecture 2, considering these and a few related problems from a `SINGULAR` point of view. We give explicit `SINGULAR` examples for several of the problems, while for others only the name of the built-in `SINGULAR` command is listed.

- Ideal membership test (Section 3.6.1) ,
- radical membership test (Section 3.6.1) ,
- elimination of variables (Section 3.6.2) ,
- intersection with free submodules (elimination of module components, Section 3.6.2) ,
- intersection of ideals and submodules (`intersect`) ,
- ideal and submodule quotients (`quotient`) ,
- saturation with respect to an ideal (`sat`) ,
- kernel of a map between affine rings (Section 3.6.3) ,
- test for algebraic dependence (Section 3.6.3) ,
- subalgebra membership test (Section 3.6.4) ,
- test for surjectivity of a map between affine rings (Section 3.6.5) ,
- Krull dimension of ideals and modules (`dim`) ,
- vector space dimension of a quotient module F/I (`vdim`) ,
- Hilbert polynomial and Hilbert series, represented as vectors of type `intvec` (`hilbPoly` from `poly.lib` and `hilb`) ,
- syzygies and free resolutions (Section 3.6.6) .

For the commands `dim` and `vdim`, we refer also to Lecture 9, Remark 9.30.

3.6.1 Ideal Membership Test

The test whether a given polynomial $f \in K[\boldsymbol{x}]$ is contained in a given ideal $I \subset K[\boldsymbol{x}]$ is arguably the most basic application of Gröbner bases. If $\mathcal{G}$ is a Gröbner basis for I, then f is contained in I iff the remainder of f on division by the elements of $\mathcal{G}$ is zero (see Remark 1.40 and Problem 2.16).

Example 3.22. Problem 1. Check ideal membership:

```
> ring R = 0, (x,y), dp;
> ideal I = x7+x5y2, y4-xy7;
> poly f1, f2 = x6y7+x3y5, x6y7+x7y2;
```

Solution. Proceed as explained above:

```
> ideal GI = groebner(I);
> reduce(f1,GI,1);     // see Example 1.39 for reduce
y5-y4
> reduce(f2,GI,1);
0
```

Alternatively, if an expression of f as a $K[\boldsymbol{x}]$-linear combination of the original generators for I is needed, apply the `lift` command which follows our solution of the Submodule Membership Problem 2.16 step by step (computing a Gröbner basis, storing the relevant syzygies, and computing a normal form). It is, thus, more involved than the `reduce` command.

```
> lift(I,f1);
   ? 2nd module lies not in the first
   ? error occurred in STDIN line 8: `lift(I,f1);  `
> matrix C = lift(I,f2);
> C;
C[1,1]=x4y22-x2y24-x3y19+xy21+y2
C[2,1]=x10y15-x6y19-x5
> f2 - C[1,1]*I[1] - C[2,1]*I[2];  // check (result must be 0)
0
```

We will return to the `lift` command in Lecture 4, Section 4.1.

Problem 2. Check the inclusion of ideals.

```
> ideal J1 = f1, f2;
> ideal J2 = f2, x5y9+x6y4;
```

Solution.

```
> reduce(J1,GI,1);     // normal form for the generators of J1
_[1]=y5-y4
_[2]=0
> size(reduce(J2,GI,1));
0
```

Recall that the `size` command applied to an `ideal` returns the number of nonzero generators. Hence, the output `0` above indicates that the ideal generated by `J2` is contained in the ideal generated by `GI`. □

As explained in Lecture 2, **radical membership** can be decided via ideal membership: if t is a slack variable, then

$$f \in \sqrt{I} \iff 1 \in \langle I, tf-1\rangle \subset K[\boldsymbol{x},t]\,.$$

Example 3.23. Problem. Check radical membership:

```
> ring R = 0, (x,y), dp;
> ideal I = maxideal(3);  // the ideal <x,y>^3
> poly f1, f2 = x, 1-x;
```

Solution. Follow the recipe given above step by step, first setting up the ring with a slack variable:

```
> ring S = 0, (x,y,t), dp;
> ideal I = imap(R,I);
> poly f1 = imap(R,f1);
> ideal Jf1 = I, t*f1-1;
> Jf1 = std(Jf1);
> reduce(1,Jf1,1);    // result is 0 iff f1 is in radical(I)
0
> poly f2 = imap(R,f2);
> ideal Jf2 = I, t*f2-1;
> Jf2 = std(Jf2);
> reduce(1,Jf2,1);
1
```

The built-in command. Directly in the ring `R`, use the `SINGULAR` command `rad_con` provided by the library `poly.lib`:

```
> LIB "poly.lib";
> rad_con(f1,I);      // result is 1 iff f is in radical(I)
1
> rad_con(f2,I);
0
```

□

3.6.2 Elimination

In Lecture 2, Proposition 2.30, we explained how to eliminate variables from a given ideal $I \subset K[x_1, \ldots, x_n]$. This requires the choice of a global elimination order. Possible choices are:

- The lexicographic order `lp`. This has the elimination property with respect to *each initial set of variables* $\{x_1, \ldots, x_k\}$, $k = 1, \ldots, n-1$.
- The product order `(dp(k),dp(n-k))`, or, more generally, any product order $(>_1, >_2)$ combining a global monomial order $>_1$ on $K[x_1, \ldots, x_k]$ and a global monomial order $>_2$ on $K[x_{k+1}, \ldots, x_n]$. Such an order is an elimination order with respect to the *specific initial set of variables* $\{x_1, \ldots, x_k\}$.
- A global monomial order $>$ with extra weight vector $\boldsymbol{w} = (w_1, \ldots, w_n) \in \mathbb{Z}^n_{\geq 0}$ (see Page 71). Such an order has the elimination property with respect to the set of variables $\{x_i \mid w_i > 0\}$.

The SINGULAR command `eliminate` uses an order with extra weight vector. If $>$ is the order of the active ring in a SINGULAR session, then `eliminate` returns a Gröbner basis for the elimination ideal with respect to the restriction of $>$ to the subring specified by the remaining variables. Note, however, that using `eliminate` might be significantly slower than using a product order `(dp(k),dp(n-k))`.

Example 3.24 (Levelt). We consider a system of 12 polynomial equations in 13 indeterminates:

```
> ring R = 0, (a,b,c,d,e,f,g,t,u,v,w,y,z), dp;
> ideal I = z2+e2-1, g2+w2+a2-1, t2+u2+b2-1, f2+v2+c2-1, y2+d2-1,
.           zw+ea, gt+wu+ab, tf+uv+bc, fy+cd, a+b+c+d+e, f+g+t+y+1,
.           u+v+w+z-1;
```

These equations define, as one can check, a curve $C = \mathrm{V}(I) \subset \mathbb{A}^{13}(\mathbb{C})$. We compute the projection of C to the yz-plane by eliminating all variables other than y, z from I. This is quite a challenge for every computer algebra system (in SINGULAR, try to compute the projection by entering directly `eliminate(I,abcdefgtuvw);`). To get a result in due time, we apply a **Hilbert driven elimination** (following the recipe given in Section 3.5):

```
> ring Rhom = 0, (a,b,c,d,e,f,g,t,u,v,w,y,z,h), dp;
> ideal I = imap(R,I);
> ideal J = homog(I,h);    // homogenize the given generators
> int aa = timer;
> ideal L = std(J);
> timer-aa;
127
> intvec H = hilb(L,1);    // assign Hilbert series
> aa = timer;
> ideal K = eliminate(J,abcdefgtuvw,H);
> timer-aa;
138
> K = subst(K,h,1);        // dehomogenize
```

The resulting projection of C to the yz-plane is a plane curve defined by a polynomial of degree 32:

```
> size(K);
1
> K[1];                    // the equation
790272y16z16-3612672y16z15+3612672y15z16-6530048y16z14-6006784y15z15
-6530048y14z16+41607168y16z13-56159232y15z14+[...]
```
$\square$

Remark 3.25. Another approach to eliminating variables, historically preceeding the one via Gröbner bases, makes use of resultants (see Grete Hermann (1926)). For more details on this approach and some examples, we refer to Lecture 6, Section 6.2. $\square$

Closely related to eliminating variables is the **elimination problem for module components**: given a free $K[\boldsymbol{x}]$-module $F = \bigoplus_{i=1}^{s} K[\boldsymbol{x}]e_i$ and a submodule $I = \langle f_1, \dots, f_r \rangle \subset F$, compute the ***k*th elimination submodule**

$$I_k := I \cap \bigoplus_{i=k+1}^{s} K[\boldsymbol{x}]e_i \subset F\,.$$

That is, compute finitely many generators for the module of vectors in I whose first k components are zero. The solution to this problem is based on the following result:

Proposition 3.26. *Let $>$ be a global monomial order on $K[\boldsymbol{x}]$, and let $\mathcal{G}$ be a Gröbner basis for I with respect to $>_{(\mathtt{c},>)}$ on F. Then, for each $k = 0, \dots, s-1$, the set*

$$\mathcal{G}_k := \mathcal{G} \cap \bigoplus_{i=k+1}^{s} K[\boldsymbol{x}]e_i$$

is a Gröbner basis for I_k with respect to $>_{(\mathtt{c},>)}$ on $\bigoplus_{i=k+1}^{s} K[\boldsymbol{x}]e_i$. In particular, $\mathcal{G}_k$ generates I_k.

Alternatively, if we are just interested in computing I_k for a fixed k, we may choose any other global **elimination order** with respect to $e_1, \dots, e_k$, that is, any global monomial order on F satisfying

$$\mathrm{L}(f) \in \bigoplus_{i=k+1}^{s} K[\boldsymbol{x}]e_i \implies f \in \bigoplus_{i=k+1}^{s} K[\boldsymbol{x}]e_i$$

for all $f \in F$. In view of Remark 3.9, this shows that there is actually no difference between the problem of eliminating variables and the problem of eliminating module components.

3.6.3 Kernel of a Ring Map

In Lecture 2, we already explained how to compute the kernel of a K-algebra homomorphism

$$\phi : K[y_1, \dots, y_m] \longrightarrow K[x_1, \dots, x_n]/I\,, \quad y_i \longmapsto \overline{f}_i = f_i + I\,,$$

via elimination:

$$\ker \phi = J \cap K[\boldsymbol{y}]\,,$$

where J is the ideal

$$J = IK[\boldsymbol{x}, \boldsymbol{y}] + \big\langle f_1 - y_1, \dots, f_m - y_m \big\rangle \subset K[\boldsymbol{x}, \boldsymbol{y}]\,.$$

Example 2.32 in Lecture 2 shows two ways of carrying this out in `SINGULAR`. We may either follow the recipe given above step by step (setting up J in $K[\boldsymbol{x}, \boldsymbol{y}]$ and eliminating), or we may apply the built-in `SINGULAR` command preimage which returns the preimage of an ideal under a given ring map (for the elimination problem above, we compute the preimage of I under the map $K[\boldsymbol{y}] \to K[\boldsymbol{x}]$ defined by the f_i). Note that similar to the eliminate command discussed in Section 3.6.2, the preimage command makes use of an order with an extra weight vector.

Being able to compute kernels of ring maps, we can, in particular, check whether a given set of elements $\overline{f}_1, \dots, \overline{f}_m$ of an affine ring $K[\boldsymbol{x}]/I$ is **algebraically dependent** over K. Indeed, this means to check whether the kernel of the map $K[\boldsymbol{y}] \to K[\boldsymbol{x}]/I$ defined by the $\overline{f}_i$ contains a nontrivial element.

Example 3.27. Problem. Use SINGULAR to check that the polynomials

$$f_1 = x_1^6 x_3^2 - x_2^6 x_3^2,\ f_2 = x_1^3 - x_2^3,\ f_3 = x_1^3 + x_2^3 \text{ and } f_4 = x_3^3$$

are algebraically dependent over $\mathbb{Q}$:

```
> ring R = 0, x(1..3), dp;
> poly f1 = x(1)^6*x(3)^2-x(2)^6*x(3)^2;
> poly f2,f3,f4 = x(1)^3-x(2)^3, x(1)^3+x(2)^3, x(3)^3;
```

Solution. Follow the recipe given above step by step (setting up ϕ and computing its kernel by using preimage):

```
> ring S = 0, y(1..4), dp;
> setring R;
> ideal zero;              // the zero ideal
> map phi = S,f1,f2,f3,f4;
> setring S;
> preimage(R,phi,zero);    // the kernel of phi
_[1]=y(2)^3*y(3)^3*y(4)^2-y(1)^3
```

The built-in command. Directly in the ring R, use the SINGULAR command algDependent provided by the library algebra.lib:

```
> LIB "algebra.lib";
> list L = algDependent(ideal(f1,f2,f3,f4));
```

At this point, SINGULAR gives an explanation on how to access what has been computed. We follow the explanation (without printing it here). On our way, we make use of the def command which allows us to assign a name to a SINGULAR object without specifying its type (the type will then be specified by SINGULAR according to the type of the object).

```
> L[1];            // first entry of L is 1 iff the polynomials are
>                  // algebraically dependent
1
. def S = L[2];    // second entry of L is a ring which contains
>                  // an ideal ker defining the algebraic relation
. setring S;
> ker;
ker[1]=y(2)^3*y(3)^3*y(4)^2-y(1)^3
```
□

3.6.4 Test for Subalgebra Membership

Problem. Given elements $\overline{f}, \overline{f}_1, \dots, \overline{f}_m$ of an affine ring $K[\boldsymbol{x}]/I$, decide whether $\overline{f}$ is contained in the subalgebra $K\big[\overline{f}_1, \dots, \overline{f}_m\big]$ of $K[\boldsymbol{x}]/I$.

Solution. Let $J = IK[\boldsymbol{x}, \boldsymbol{y}] + \langle f_1 - y_1, \dots, f_m - y_m \rangle \subset K[\boldsymbol{x}, \boldsymbol{y}]$, let $>$ be a global elimination order on $K[\boldsymbol{x}, \boldsymbol{y}]$ with respect to $\boldsymbol{x}$, and let $\mathcal{G}$ be a Gröbner basis for J with respect to $>$. Then $\overline{f} \in K\big[\overline{f}_1, \dots, \overline{f}_m\big]$ iff the remainder h of f on division by the elements of $\mathcal{G}$ is in $K[\boldsymbol{y}]$. The remainder h describes, then, how to write $\overline{f}$ as a polynomial in $\overline{f}_1, \dots, \overline{f}_m$.

Example 3.28. Problem. Use SINGULAR to check whether $x_1^6 x_2^6 - x_1^6 x_3^6$ is contained in the subalgebra $\mathbb{Q}\big[x_1^3 x_2^3 - x_1^3 x_3^3, x_1^3 x_2^3 + x_1^3 x_3^3\big]$ of $\mathbb{Q}\big[x_1, x_2, x_3\big]$:

```
> ring R = 0, x(1..3), dp;
> poly f = x(1)^6*x(2)^6-x(1)^6*x(3)^6;
> poly f1 = x(1)^3*x(2)^3-x(1)^3*x(3)^3;
> poly f2 = x(1)^3*x(2)^3+x(1)^3*x(3)^3;
```

Solution. Follow the recipe given above step by step:

```
> ring S = 0, (x(1..3),y(1..2)), (dp(3),dp(2));
> ideal J = imap(R,f1)-y(1), imap(R,f2)-y(2);
> ideal G = groebner(J);
> reduce(imap(R,f),G);
y(1)*y(2)
```

From the output, we read that $f = f_1 f_2 \in \mathbb{Q}[f_1, f_2]$.

The built-in command. Directly in the ring R, use the SINGULAR command algebra_containment from algebra.lib:

```
> LIB "algebra.lib";
> algebra_containment(f,ideal(f1,f2));
// y(1)*y(2)
1
```

The return value 1 indicates that $f \in \mathbb{Q}[f_1, f_2]$, while the displayed polynomial y(1)*y(2) stands for the algebraic relation $f_1 f_2 - f = 0$.

Calling algebra_containment with the additional parameter 1 will allow us to access the polynomial:

```
> def L = algebra_containment(f,ideal(f1,f2),1);
```

Again, SINGULAR displays an explanation which we follow without printing it here:

```
> def S = L[2];
> setring S;
> check;              // polynomial defining the algebraic relation
y(1)*y(2)
```

Note that the SINGULAR library algebra.lib provides the alternative command inSubring for the subalgebra membership problem. This command is based on a slightly different algorithm (see Greuel and Pfister (2002), Section 1.8.11). □

3.6.5 Test for Surjectivity of a Ring Map

Problem. Given a map $\phi : K[\boldsymbol{y}] \to K[\boldsymbol{x}]/I$, $y_i \mapsto \overline{f}_i$, find out whether ϕ is surjective. That is, check whether $\overline{x}_1, \dots, \overline{x}_n$ are contained in the subalgebra $K\left[\overline{f}_1, \dots, \overline{f}_m\right] \subset K[\boldsymbol{x}]/I$.

Solution. Set $J = IK[\boldsymbol{x}, \boldsymbol{y}] + \langle f_1 - y_1, \dots, f_m - y_m\rangle \subset K[\boldsymbol{x}, \boldsymbol{y}]$, and compute the reduced Gröbner basis $\mathcal{G}$ for J using a global elimination order with respect to $\boldsymbol{x}$. Then $\overline{x}_i \in K\left[\overline{f}_1, \dots, \overline{f}_m\right]$ iff $\mathcal{G}$ contains an element of type $x_i - h_i(\boldsymbol{y})$, for $i = 1, \dots, n$.

Example 3.29. Problem. Let $\phi : \mathbb{Q}[x,y,z] \to \mathbb{Q}[a,b,c]/\langle c - b^3\rangle$ be induced by $x \mapsto 2a + b^6$, $y \mapsto 7b - a^2$, $z \mapsto c^2$. Check that ϕ is surjective using SINGULAR.

Solution. Follow the recipe given above step by step:

```
> ring C = 0, (a,b,c,x,y,z), (dp(3),dp);
> ideal J = c-b3, 2a+b6-x, 7b-a2-y, c2-z;
> option(redSB);
> simplify(groebner(J),1);    // the reduced Groebner basis for J
_[1]=x12-12x11z+66x10z2-220x9z3+495x8z4-792x7z5+[...]
_[2]=c-1/21952x6+3/10976x5z-15/21952x4z2+[...]
_[3]=b-1/28x2+1/14xz-1/28z2-1/7y
_[4]=a-1/2x+1/2z
```

The built-in command. Use the command is_surjective from algebra.lib:

```
> LIB "algebra.lib";
> ring B = 0, (x,y,z), dp;
> ring A = 0, (a,b,c), dp;
> qring Q = groebner(c-b3);    // quotient ring
> map psi = B, 2a+b6, 7b-a2, c2;
> is_surjective(psi);
1
```
□

Note that algebra.lib also provides the command is_bijective.

3.6.6 Syzygies and Free Resolutions

Consider the free $K[\boldsymbol{x}]$-module $F = K[\boldsymbol{x}]^s$ with its canonical basis, and let $f_1, \dots, f_r \in F \setminus \{0\}$ be polynomial vectors. Recall from Remark 1.44 how to compute the syzygies on $f_1, \dots, f_r$ using Schreyer's algorithm:

- Compute a Gröbner basis $f_1, \dots, f_r, f_{r+1}, \dots, f_{r'}$ with Buchberger's algorithm. On your way, store all syzygies on the elements of the Gröbner basis defined by a standard expression in Buchberger's test; these syzygies generate all syzygies on the elements of the Gröbner basis.
- The syzygies obtained from a division leading to a new generator f_k in Buchberger's test allow us to express f_k in terms of $f_1, \dots, f_{k-1}$. Successively substituting these expressions for f_k into each relation obtained from a division with remainder zero in the test, we get the syzygies on the original generators $f_1, \dots, f_r$.

The SINGULAR command sres is based on Schreyer's algorithm (see Remark 3.33). However, sres is implemented such that it requires as input a Gröbner basis (see Example 3.35). Hence, it can only be used to compute the syzygies on the elements of a Gröbner basis (and not those on an arbitrary set of generators).

An alternative method for computing syzygies is based on the following easy observation. View $f_1, \dots, f_r \in F = K[\boldsymbol{x}]^s$ as column vectors with s entries, and consider the $(s+r) \times r$ matrix

$$\begin{pmatrix} f_1 & f_2 & \dots & f_r \\ 1 & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 1 \end{pmatrix} .$$

Then the vector ${}^t(g_1, \dots, g_r) \in K[\boldsymbol{x}]^r$ is a syzygy on $f_1, \dots, f_r$ iff the extended vector ${}^t(\mathbf{0}, g_1, \dots, g_r) \in K[\boldsymbol{x}]^{s+r}$ is a $K[\boldsymbol{x}]$-linear combination of the columns of this matrix. In this way, computing syzygies amounts to eliminating module components (see Proposition 3.26). The SINGULAR command syz makes use of this idea.

Example 3.30. We show syz at work:

```
> ring R = 0, (w,x,y,z), dp;
> poly f1, f2, f3 = y2-xz, xy-wz, x2z-wyz;
> ideal I = f1, f2, f3;
> module phi2 = syz(I);
> print(phi2);
x, wz,
-y,-xz,
1, y
> size(syz(phi2));  // we check that there are no higher syzygies
0
```

□

By successively computing syzygies, starting from $f_1, \dots, f_r$ and using either Schreyer's algorithm or the alternative method, we eventually obtain a free resolution

$$F_1 \xleftarrow{\varphi_2} F_2 \xleftarrow{\varphi_3} F_3 \xleftarrow{\varphi_4} \cdots$$

of the submodule $I = \langle f_1, \ldots, f_r \rangle \subset F$ and, thus, a free resolution

$$F_0 = F \xleftarrow{\varphi_1 = (f_1, \ldots, f_r)} F_1 \xleftarrow{\varphi_2} F_2 \xleftarrow{\varphi_3} F_3 \xleftarrow{\varphi_4} \cdots$$

of the quotient module $M = F/I$, with syzygy matrices φ_i.

If $f_1, \ldots, f_r$ are homogeneous, both ways of computing syzygies yield graded free resolutions. However, since we compute at each stage a Gröbner basis for the syzygy module, the resulting resolutions are usually far from being minimal. This is already evident from Example 3.30 above.

To detect and get rid of superfluous generators in a given set of generators, we need to know a syzygy matrix of the given generators. Minimizing the given generators amounts, then, to Gaussian elimination applied to the syzygy matrix.

Example 3.31. Consider the graded free resolution

$$R \xleftarrow{\varphi_1 = (f_1, f_2, f_3)} R^2(-2) \oplus R(-3) \xleftarrow{\varphi_2} R(-3) \oplus R(-4) \longleftarrow 0$$

computed in Example 3.30. The entry 1 of the syzygy matrix φ_2 indicates that f_1, f_2, f_3 do not form a minimal set of generators for I. In fact, from the first column of φ_2, we see that $f_3 = -xf_1 + yf_2$. Gaussian elimination applied to φ_2 yields a commutative diagram

$$\begin{tikzcd}
R \arrow[d, equal] & R^2(-2) \oplus R(-3) \arrow[l, "\varphi_1"'] \arrow[d, "\alpha_1"] & R(-3) \oplus R(-4) \arrow[l, "\varphi_2"'] \arrow[d, "\alpha_2"] & 0 \arrow[l] \\
R & R^2(-2) \oplus R(-3) \arrow[l, "\widetilde{\varphi}_1"'] & R(-3) \oplus R(-4) \arrow[l, "\widetilde{\varphi}_2"'] & 0, \arrow[l]
\end{tikzcd}$$

where

$$\alpha_1 = \begin{pmatrix} 1 & 0 & -x \\ 0 & 1 & y \\ 0 & 0 & 1 \end{pmatrix}, \quad \alpha_2 = \begin{pmatrix} 1 & y \\ 0 & 1 \end{pmatrix}, \quad \widetilde{\varphi}_1 = (f_1, f_2, 0), \quad \widetilde{\varphi}_2 = \begin{pmatrix} 0 & -f_2 \\ 0 & f_1 \\ 1 & 0 \end{pmatrix}.$$

Cancelling the superfluous part

$$0 \longleftarrow R(-3) \xleftarrow{1} R(-3) \longleftarrow 0$$

in the second row, we a get the minimal free resolution of R/I:

$$R \xleftarrow{(f_1, f_2)} R^2(-2) \xleftarrow{\binom{-f_2}{f_1}} R(-4) \longleftarrow 0\,. \qquad \square$$

Using the alternative method for computing free resolutions, minimization can be done directly at each stage, minimizing each set of generators as soon as the syzygies on it have been computed. In contrast, Schreyer's algorithm requires that at each stage we work with the syzygies on a Gröbner basis. Thus, minimization has to be done after the whole resolution has been computed.

A variant of Schreyer's algorithm due to La Scala (see La Scala and Stillman (1998)) uses a "horizontal" strategy, proceeding degree by degree (not syzygy module by syzygy module). In this way, direct minimization is possible.

A horizontal Hilbert driven algorithm for computing free resolutions has been described by Capani, De Dominicis, Niesi and Robbiano (1997).

Remark 3.32. Schreyer's algorithm is more conceptual than the alternative method. As already pointed out in Lecture 1, Remark 1.45, it allows one to give a constructive proof of Hilbert's syzygy theorem. Also, if I is a graded submodule of a graded free $K[\boldsymbol{x}]$-module, it follows from applying Schreyer's algorithm to both I and $\mathrm{L}(I)$ that $\beta_{ij}(F/I) \leq \beta_{ij}(F/\mathrm{L}(I))$ for all i,j (see Decker and Schreyer (2006)). This fact can be used to expedite the computation of free resolutions via the alternative method. □

Remark 3.33 (`SINGULAR` Commands for Computing Free Resolutions I: Polynomial Rings). The following commands are available:

Command	Implemented Method	Applies to
`nres`	alternative method	ideals, modules
`mres`	alternative method	ideals, modules
`sres`	Schreyer	ideals, modules
`lres`	La Scala	homogeneous ideals
`hres`	Hilbert driven	homogeneous ideals

Each command requires two input parameters, one of type `ideal` or `module`, and one of type `int`. The integer, say k, has to be nonnegative. If $k > 0$, the free resolution is computed up to stage k; if $k = 0$, it is computed up to stage $n + 1$, where n is the number of variables in the active ring. The result of such a computation has its own `SINGULAR` type, namely `resolution`. To give a first example, we continue the session from Example 3.30:

```
> resolution FI = nres(I,0);
> typeof(FI[1]);          // 'typeof' displays type of given object
ideal
> print(FI[1]);
y2-xz,
xy-wz
x2z-wyz
> typeof(FI[2]);
module
> print(FI[2]);
x, wz,
-y,-xz,
1, y
```

It is hard to predict a priori which of the commands for computing free resolutions is best suited for a given problem. □

Before giving an example which, in particular, illustrates the difference between `nres` and `mres`, we explain the use of the `betti` command:

Remark 3.34 (Graded Betti Numbers). Let I be a graded submodule of a graded free module F, and suppose that a graded free resolution `FI` of I has already been computed. Then the graded Betti numbers of F/I can be directly read off, even if `FI` is not yet minimal. In SINGULAR, these numbers will be displayed as response to entering `print(betti(FI),"betti");`. That is, typing `print(betti(FI),"betti");` will make SINGULAR print numerical information on the *minimal* free resolution of F/I (and not on the resolution which has been computed). For instance, for the ideal I considered in Example 3.30 and Remark 3.33, we obtain:

```
> print(betti(FI),"betti");
           0     1     2
------------------------
    0:     1     -     -
    1:     -     2     -
    2:     -     -     1
------------------------
total:     1     2     1
```

To get numerical information on the *computed* resolution, type

```
> print(betti(FI,0),"betti");
           0     1     2
------------------------
    0:     1     -     -
    1:     -     2     1
    2:     -     1     1
------------------------
total:     1     3     2
```

Similarly, to visualize the degrees of the generators of a given homogeneous ideal (submodule) I, enter `print(betti(I,0),"betti");`. □

In the following example, we show the behavior of `nres`, `mres`, and `sres`, and we introduce the commands `prune` and `minres`.

Example 3.35. To begin with, we define the ring `R` and the module `I` which we already considered in Section 3.4:

```
> 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];
```

```
> print(I);
xz,-yz2,y2z,y3, -z3,-yz2,0,
0, y2,  0,  0,  yz, y2,  0,
-w,0,   -z2,-yz,0,  0,   -wy2+xz2,
-1,-w,  0,  -x, 0,  -w,  -y2,
0, 0,   -x, 0,  -w, 0,   x2
```

Recall from Section 3.4 that we may think of I as a graded submodule of the graded free module $F = R \oplus R(-1)^2 \oplus R(-2)^2$:

```
> homog(I);
1
> attrib(I,"isHomog");
0,1,1,2,2
```

Applying `nres` to compute a free resolution of I, the given generators remain untouched, but for each subsequent syzygy module, a minimal set of generators will be computed:

```
> resolution FInres = nres(I,0);
> print(betti(FInres,0),"betti");
           0     1     2
------------------------
    0:     1     -     -
    1:     2     1     1
    2:     2     5     1
    3:     -     1     1
------------------------
total:     5     7     3
> print(FInres[1]);   // the given generators
xz,-yz2,y2z,y3, -z3,-yz2,0,
0, y2,  0,  0,  yz, y2,  0,
-w,0,   -z2,-yz,0,  0,   -wy2+xz2,
-1,-w,  0,  -x, 0,  -w,  -y2,
0, 0,   -x, 0,  -w, 0,   x2
> print(FInres[2]);   // display syzygies on the given generators
0, y2,0,
-1,0, xz,
0, -x,wy,
0, 0, -wz,
0, 0, -xy,
1, 0, 0,
0, -1,0
> size((FInres[3]));
0
```

The nonzero constant entries of the syzygy matrix `FInres[2]` indicate that in our example, the given generators do not form a minimal set of generators. Using `mres`, the given set of generators will be minimized before computing the resolution:

```
> resolution FImres = mres(I,0);
> print(betti(FImres,0),"betti");
           0     1     2
------------------------
    0:     1     -     -
    1:     2     1     -
    2:     2     4     -
    3:     -     -     1
------------------------
total:     5     5     1
> print(FImres[1]);   // the new generators
xz,z3, yz2,y2z,y3,
0, -yz,-y2,0,  0,
-w,0,  0,  -z2,-yz,
-1,0,  w,  0,  -x,
0, w,  0,  -x, 0
> print(FImres[2]);   // display syzygies on the new generators
0,
xy,
-xz,
wy,
-wz
```

Thus, considering I as a submodule of $F = R \oplus R(-1)^2 \oplus R(-2)^2$ as above, its minimal free resolution is $R(-2) \oplus R(-3)^4 \leftarrow R(-5) \leftarrow 0$.

Note, however, that there is still a nonzero constant entry of `FImres[1]` (the columns of `FImres[1]` form the original set of generators for I). Hence, $F \leftarrow R(-2) \oplus R(-3)^4 \leftarrow R(-5) \leftarrow 0$ is not the minimal free resolution of the module $M = F/I$. More precisely, the images of the canonical basis vectors of F under $M \leftarrow F$ do not generate M minimally. To obtain the minimal free resolution of M, we first have to compute a presentation

$$0 \longleftarrow M \xleftarrow{\varphi_0} F_0 \xleftarrow{\widetilde{\varphi}_1} \widetilde{F}_1$$

such that φ_0 corresponds to a minimal set of generators for M. This is done by Gaussian elimination. The corresponding SINGULAR command is `prune`:

```
> module PI = prune(I);
> print(betti(PI,0),"betti");
           0     1
------------------
    0:     1     -
    1:     2     -
    2:     1     5
    3:     -     1
------------------
total:     4     6
```

```
> print(PI);
wxz+yz2,-y2z,-y3+x2z,z3, wxz+yz2,xy2z,
-y2,    0,   0,      -yz,-y2,    0,
-w2,    z2,  -wx+yz, 0,  -w2,    -xz2,
0,      x,   0,      w,  0,      -x2
> resolution FPImres = mres(PI,0);
> print(betti(FPImres,0),"betti");
           0     1     2
------------------------
    0:     1     -     -
    1:     2     -     -
    2:     1     4     -
    3:     -     -     1
------------------------
total:     4     4     1
> print(FPImres[1]);
z3, y2z,wxz+yz2,y3-x2z,
-yz,0,  -y2,    0,
0,  -z2,-w2,    wx-yz,
w,  -x, 0,      0
> print(FPImres[2]);
xy,
wy,
-xz,
-wz
```

Thus, the minimal free resolution of M is of type

$$R \oplus R(-1)^2 \oplus R(-2) \longleftarrow R(-3)^4 \longleftarrow R(-5) \longleftarrow 0\,.$$

Next, we apply `sres` to `PI`. As already remarked, this command requires a Gröbner basis as input. This is checked by `SINGULAR`:

```
> resolution FPIsres = sres(PI,0);
   ? ideal not a standard basis
   ? error occurred in STDIN line 27:
     ‘resolution FPIsres = sres(PI,0);‘
> resolution FPIsres = sres(groebner(PI),0);
> print(betti(FPIsres,0),"betti");
           0     1     2     3     4
------------------------------------
    0:     1     -     -     -     -
    1:     2     -     -     -     -
    2:     1     4     1     -     -
    3:     -     1     5     1     -
    4:     -     4     2     3     1
    5:     -     1     3     1     -
------------------------------------
total:     4    10    11     5     1
```

As the reader might have expected, the resolution is far from being minimal. In fact, it is even longer than the minimal free resolution.

Applying the command `minres` to a given resolution `FI` of `I`, all syzygy matrices of `I` will be minimized. Note, however, that `minres` does not prune `I`. That is, the resolution obtained when entering `minres(FI)` is isomorphic to that obtained when entering `mres(I)`:

```
> resolution FInresmin = minres(FInres);
> print(betti(FInresmin,0),"betti");
           0     1     2
------------------------
    0:     1     -     -
    1:     2     1     -
    2:     2     4     -
    3:     -     -     1
------------------------
total:     5     5     1
> resolution FPIsresmin = minres(FPIsres);
> print(betti(FPIsresmin,0),"betti");
           0     1     2
------------------------
    0:     1     -     -
    1:     2     -     -
    2:     1     4     -
    3:     -     -     1
------------------------
total:     4     4     1
```

□

Remark 3.36 (SINGULAR Commands for Computing Free Resolutions II: Quotient Rings and Local Rings).

(1) The commands `syz`, `nres`, `mres`, and `sres` also work over quotient rings of polynomial rings. Note, however, that modules over such a ring may not have a free resolution of finite length. See Example 4.9.

(2) Recall from Remark 1.17 that it makes sense to speak of minimal free resolutions over local Noetherian rings, too (in general, as in part (1) above, a free resolution of finite length may not exist). In Lecture 9, we will extend the concept of Gröbner bases to the ring $K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle}$ which is obtained by localizing $K[\boldsymbol{x}]$ at the maximal ideal $\langle \boldsymbol{x} \rangle$ corresponding to the origin of affine space. Adapting Buchberger's algorithm to local monomial orders, we will be able to compute Gröbner bases and syzygies over $K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle}$. In SINGULAR, the commands `syz`, `nres`, `mres`, and `sres` also work over $K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle}$ and quotient rings thereof. Note that in theoretical terms, Schreyer's algorithm shows as for the polynomial ring itself that Hilbert's syzygy theorem holds over $K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle}$: every finitely generated $K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle}$-module has a free resolution of length at most n. □

3.7 Gröbner Bases over Noncommutative Algebras

The `SINGULAR` kernel component `PLURAL`[4] allows us to perform computations over a large class of noncommutative algebras to which we refer as GR-algebras[5]. In Appendix A, we will discuss one application of this in algebraic geometry, namely the computation of sheaf cohomology via free resolutions over the exterior algebra. In this section, we briefly sketch the theoretical background of `PLURAL`, illustrating the usage of its basic commands by some small pieces of `SINGULAR` code.

GR-algebras are obtained by imposing specific relations on the free noncommutative algebra on $x_1, \dots, x_n$. We denote this free algebra by $K\langle \boldsymbol{x}\rangle = K\langle x_1, \dots, x_n\rangle$. That is, $K\langle \boldsymbol{x}\rangle$ is the K-algebra with K-vector space basis

$$\mathscr{B} = \left\{x_{i_1}x_{i_2}\cdots x_{i_\nu} \;\middle|\; \nu \in \mathbb{N},\, 1 \le i_\ell \le n \text{ for all } \ell\right\},$$

where multiplying two elements of $\mathscr{B}$ means to concatenate these elements. Note that $K\langle \boldsymbol{x}\rangle$ carries a natural grading which is defined by assigning the degree ν to $x_{i_1}x_{i_2}\cdots x_{i_\nu}$.

We refer to the elements of $\mathscr{B}$ as **words** in (the **letters**) $x_1, \dots, x_n$. Moreover, we set

$$\mathscr{M} := \left\{\boldsymbol{x}^\alpha = x_1^{\alpha_1}\cdots x_n^{\alpha_n} \;\middle|\; \alpha = (\alpha_1, \dots, \alpha_n) \in \mathbb{N}^n\right\} \subset \mathscr{B}.$$

Since $\mathscr{M}$ can be identified with the set of monomials in $K[\boldsymbol{x}]$, each monomial order $>$ on $K[\boldsymbol{x}]$ induces a total order on $\mathscr{M}$ which we again denote by $>$. It, thus, makes sense to speak of the **leading term** $\mathrm{L}(h) = \mathrm{L}_>(h)$ of a K-linear combination h of words in $\mathscr{M}$.

Definition 3.37 (G-algebra, Preliminary Version). A **G-algebra** R is the quotient of $K\langle \boldsymbol{x}\rangle$ by a two-sided ideal J_0 generated by elements of type

$$x_jx_i - c_{ij}x_ix_j - h_{ij}\,, \quad 1 \le i < j \le n\,, \tag{3.1}$$

where the c_{ij} are nonzero scalars, and where the h_{ij} are K-linear combinations of words in $\mathscr{M}$. Further, we require that

(G1) $c_{ik}c_{jk}h_{ij}x_k - x_kh_{ij} + c_{jk}x_jh_{ik} - c_{ij}h_{ik}x_j + h_{jk}x_i - c_{ij}c_{ik}x_ih_{jk} = 0$

for all $1 \le i < j < k \le n$, and that

(G2) there exists a global monomial order $>$ on $K[\boldsymbol{x}]$ such that $x_ix_j > \mathrm{L}(h_{ij})$ for all i, j. □

Each global monomial order on $K[\boldsymbol{x}]$ satisfying (G2) is called an **admissible monomial order** for R. We refer to the elements of $\mathscr{M}$ as **monomials** in R and to a scalar times a monomial as a **term** in R.

[4] The name `PLURAL` results from the obvious wordplay.
[5] GR stands for Gröbner-ready.

Remark 3.38. The "rewriting relations" (3.1) together with (G2) imply that each element of R can be represented by a K-linear combination of monomials. More precisely, successively rewriting[6] a word $w = x_{i_1}x_{i_2}\cdots x_{i_\nu}$ according to the relations (3.1), we get

$$w \equiv c\boldsymbol{x}^\alpha + h \bmod J_0\,,$$

where $c \in K \setminus \{0\}$, $\boldsymbol{x}^\alpha$ is the monomial obtained by rearranging the letters of w, and h is a K-linear combination of monomials such that $\boldsymbol{x}^\alpha > \mathrm{L}(h)$. □

Definition 3.39 (G-algebra, Final Version). A **G-algebra**[7] is defined as in Definition 3.37, with condition (G1) being replaced by the weaker condition

(G1') Successively rewriting[6] the left-hand side of (G1) according to the relations (3.1), we get 0. □

The condition (G1') guarantees that the representation discussed in Remark 3.38 above is uniquely determined. Indeed, for $i < j < k$, it guarantees that the result obtained by rewriting $x_kx_jx_i$ in terms of monomials does not depend on whether we first apply the rewriting relation for the pair (j, k) or that for the pair (i, j). As a consequence, we get that $\mathscr{M}$ is a K-vector space basis for R (see Levandovskyy (2005), Chapter 1, Section 2).

In what follows, we explain how to define and compute Gröbner bases over G-algebras. This allows us to compute over G-algebras and quotients thereof.

Definition 3.40. A **GR-algebra** A is the quotient $A = R/J$ of a G-algebra R by a two-sided ideal $J \subset R$. □

To implement an explicitly given G-algebra R in SINGULAR proceed as follows (the notations are as in Definition 3.37):

Step 1. Define a `ring` which implements $K[\boldsymbol{x}]$ with a global monomial order that is admissible for R.

Step 2. Define a `matrix C[n][n]` with entries `C[i,j]` $= c_{ij}$ of type `number` (for $i \geq j$, set $c_{ij} = 0$).

Step 3. Define a `matrix H[n][n]` with entries `H[i,j]` $= h_{ij}$ of type `poly` (for $i \geq j$, set $h_{ij} = 0$).

Step 4. Enter `ncalgebra(C,H);` to activate the rewriting relations (3.1). The active basering is now an implementation of R. It is accessible under the name of the `ring` defined in Step 1.

[6] Each single rewriting step consists of replacing a product of letters x_jx_i, $i < j$, occurring in some word of the current intermediate result by the K-linear combination of monomials $c_{ij}x_ix_j + h_{ij}$. Extending the "monomial" order on $\mathscr{M}$ to an appropriate order $>$ on the words in $K\langle\boldsymbol{x}\rangle$ such that $x_jx_i > x_ix_j$ if $i < j$, we may regard this step as a single division step. Then **"successively rewriting"** means nothing but applying an indeterminate division algorithm (see Levandovskyy (2005), Chapter 1, Section 2).

[7] Here, we follow the notation used in the SINGULAR manual.

Note that the `ncalgebra` command does not check whether `C` and `H` satisfy one of the conditions (G1) or (G1'). To check (G1'), use the command `ndcond` from `nctools.lib`.

Having implemented R, we can implement a GR-algebra $A = R/J$ by using the `qring` command. Similar to the commutative case, this requires the computation of a Gröbner basis for J in R (see Definition 3.42). The command for computing a Gröbner basis for the two-sided ideal generated by given elements of R is `twostd` (see page 108 for this command).

Example 3.41 (Ring Definitions III: GR-Algebras). We show how to implement the Weyl algebra and the exterior algebra. The Weyl algebra is a G-algebra, while for the exterior algebra, we have to make use of the `qring` command. In both cases, it follows directly from the definitions that all global monomial orders are admissible. In our implementations, we choose $>_{\texttt{dp}}$.

(1) The elements of the **Weyl algebra** D_n are K-linear differential operators on $K[\boldsymbol{x}]$. Its generators are $x_1, \dots, x_n, \partial_1, \dots, \partial_n$, where we think of x_i as the operator which multiplies a polynomial by x_i and where applying ∂_i means to take the partial derivative with respect to x_i. To obtain D_n from the free algebra $K\langle x_1, \dots, x_n, \partial_1, \dots, \partial_n\rangle$, we have to impose the relations between the generating operators, that is, we have to factor out the two-sided ideal generated by the elements

$$\partial_i x_i - x_i\partial_i - 1\,, \quad \partial_i x_j - x_j\partial_i\,, \quad 1 \le i \ne j \le n\,,$$
$$\partial_j\partial_i - \partial_i\partial_j\,, \quad x_j x_i - x_i x_j\,, \quad 1 \le i < j \le n\,.$$

The following `SINGULAR` session implements D_3 over $\mathbb{Q}$:

```
> ring D3 = 0, (x(1..3),d(1..3)), dp;
> int i,j;
> matrix C[6][6];
> for (i=1; i<=6; i++) { for (j=i+1; j<=6; j++) { C[i,j] = 1; } }
> matrix H[6][6];
> H[1,4] = 1;  H[2,5] = 1;  H[3,6] = 1;
> ncalgebra(C,H);
```

Alternatively, apply the built-in command `Weyl` from the library `nctools.lib`:

```
> LIB "nctools.lib";
> ring D3 = 0, (x(1..3),d(1..3)), dp;
> Weyl();
```

Using the `basering` command to make `SINGULAR` display information on the active ring, we get in both cases:

```
> basering;
//   characteristic : 0
//   number of vars : 6
//        block   1 : ordering dp
//                    : names    x(1) x(2) x(3) d(1) d(2) d(3)
//        block   2 : ordering C
//   noncommutative relations:
//    d(1)x(1)=x(1)*d(1)+1
//    d(2)x(2)=x(2)*d(2)+1
//    d(3)x(3)=x(3)*d(3)+1
```

(2) The **exterior algebra** E_n on the K-vector space with basis $x_1, \ldots, x_n$ is obtained from $K\langle \boldsymbol{x} \rangle$ by imposing skew-commutativity, that is, by factoring out the two-sided ideal generated by the elements

$$x_j x_i + x_i x_j\,, \quad x_i^2\,, \quad 1 \le i < j \le n\,.$$

The following SINGULAR session implements E_3 over $\mathbb{Q}$:

```
> ring R = 0, x(1..3), dp;
> int i,j;
> matrix C[3][3];
> for (i=1; i<=3; i++) { for (j=i+1; j<=3; j++) { C[i,j] = -1; } }
> matrix H[3][3];
> ncalgebra(C,H);
> ideal Q = x(1)^2, x(2)^2, x(3)^2;
> Q = twostd(Q);  // compute Groebner basis for two-sided ideal
> qring E3 = Q;
```

Alternatively, use the built-in command `Exterior` from `nctools.lib`:

```
> LIB "nctools.lib";
> ring R = 0, x(1..3), dp;
> def E3 = Exterior();
> setring E3;
```

In both cases, we get:

```
> basering;
//   characteristic : 0
//   number of vars : 3
//        block   1 : ordering dp
//                  : names    x(1) x(2) x(3)
//        block   2 : ordering C
//   noncommutative relations:
//    x(2)x(1)=-x(1)*x(2)
//    x(3)x(1)=-x(1)*x(3)
//    x(3)x(2)=-x(2)*x(3)
// quotient ring from ideal
_[1]=x(3)^2
_[2]=x(2)^2
_[3]=x(1)^2
```

□

We now turn to Gröbner bases over G-algebras. In formulating the basic definitions and results, we essentially work with left ideals (the case of right ideals is completely analogous).

Let R be a G-algebra, and let $>$ be an admissible monomial order for R. According to Remark 3.38, we may think of the elements of R as K-linear combinations of monomials. In particular, it makes sense to speak of the **leading term** $\mathrm{L}(f) = \mathrm{L}_>(f)$ of an element $f \in R$.

Definition 3.42. (1) We say that a nonzero term $a\boldsymbol{x}^\alpha$ in R is **divisible** by a nonzero term $b\boldsymbol{x}^\beta$ in R if $\beta_i \leq \alpha_i$ for each $1 \leq i \leq n$.

(2) A finite subset $\mathcal{G} = \{f_1, \dots, f_r\}$ of a left ideal $I \subset R$ is called a **(left) Gröbner basis for *I*** with respect to $>$ if for each $f \in I \setminus \{0\}$, the leading term of f is divisible by the leading term of some f_i, $1 \leq i \leq r$.

(3) We say that a finite subset of R is a **left Gröbner basis** if it is a Gröbner basis for the left ideal it generates.

(4) If $I \subset R$ is a two-sided ideal, the monomials in R which are not divisible by the leading term of some $f \in I$ are called **standard monomials** (for I, with respect to $>$). □

Remark 3.43. (1) Each left ideal $I \subset R$ has a left Gröbner basis due to Gordan's lemma. In particular, R is **left Noetherian**, that is, each left ideal is finitely generated.

(2) If $I \subset R$ is a two-sided ideal, each left Gröbner basis for I is also a right Gröbner basis for I. Indeed, the divisibility of terms in R is independent of "left" or "right".

(3) Given a two-sided ideal $I \subset R$, the residue classes of the standard monomials for I form a K-vector space basis for R/I. As in the commutative case, this basis may be obtained by computing a Gröbner basis for I.

(4) If a nonzero term $a\boldsymbol{x}^\alpha$ in R is divisible by a nonzero term $b\boldsymbol{x}^\beta$, there exist uniquely determined $c \in K \setminus \{0\}$ and $h \in R$ such that $a\boldsymbol{x}^\alpha = c\boldsymbol{x}^{\alpha-\beta} \cdot b\boldsymbol{x}^\beta + h$, where $\boldsymbol{x}^\alpha > \mathrm{L}(h)$. Moreover, there is an obvious algorithm for computing c and h. This allows us to mimic the steps of the division algorithm in $K[\boldsymbol{x}]$. As a result, we get an algorithm for **left division with remainder** in R. In particular, we get a division theorem extending Theorem 1.38 in Lecture 1. Further, Remark 1.40 applies accordingly: each left Gröbner basis for a left ideal $I \subset R$ generates I (as a left ideal), and we can speak of **left normal forms** of elements of R mod I. □

If F is the free R-module R^s with its canonical left and right module structures and with its canonical basis $e_1, \dots, e_s$, then we can define monomials and terms in F in the usual way. A monomial order on F is nothing but a monomial order on the free $K[\boldsymbol{x}]$-module with basis $e_1, \dots, e_s$. Such an order is called **admissible** if the induced order on $K[\boldsymbol{x}]$ is admissible for R. Definition 3.42 and Remark 3.43 above apply accordingly to left (respectively two-sided) submodules of F.

Definition 3.44. Let $f, g \in F \setminus \{0\}$. If $\mathrm{L}(f)$ and $\mathrm{L}(g)$ involve the same basis element, say $\mathrm{L}(f) = a\boldsymbol{x}^\alpha e_k$, $\mathrm{L}(g) = b\boldsymbol{x}^\beta e_k$, we define the **left S-polynomial** of f and g to be

$$\mathrm{S}(f,g) = c_1 b\boldsymbol{x}^{\gamma-\alpha} \cdot f - c_2 a\boldsymbol{x}^{\gamma-\beta} \cdot g\,,$$

where $\gamma_i = \max\{\alpha_i, \beta_i\}$, $i = 1, \ldots, n$, and where $c_1, c_2 \in K \setminus \{0\}$ are the unique coefficients appearing when dividing $\boldsymbol{x}^\gamma$ by $\boldsymbol{x}^\alpha$, $\boldsymbol{x}^\beta$:

$$\boldsymbol{x}^\gamma = c_1\boldsymbol{x}^{\gamma-\alpha} \cdot \boldsymbol{x}^\alpha + h_1 = c_2\boldsymbol{x}^{\gamma-\beta} \cdot \boldsymbol{x}^\beta + h_2, \quad \boldsymbol{x}^\gamma > \mathrm{L}(h_1), \mathrm{L}(h_2).$$

If $\mathrm{L}(f)$ and $\mathrm{L}(g)$ involve different basis elements, we set $\mathrm{S}(f,g) = 0$. □

With this notion of S-polynomials, the analogs to Buchberger's criterion and Schreyer's result on syzygies in Lecture 1 can be proved for G-algebras. In particular, we have variants of the algorithms of Buchberger and Schreyer for computing left Gröbner bases and syzygies. Remark 1.47 on reduced Gröbner bases applies accordingly.

The `PLURAL` implementation of Buchberger's algorithm is accessible via the `std` command. It allows one to work over the large class of GR-algebras and is tuned for that general purpose.[8]

If we apply `std` to $f_1, \ldots, f_r \in F$, `SINGULAR` computes a left Gröbner basis for the left submodule of F generated by $f_1, \ldots, f_r$. To obtain a right Gröbner basis for the right submodule generated by $f_1, \ldots, f_r$, perform the Gröbner basis computation over the **opposite algebra** of R, which is defined by reversing the order of the multiplication operation of R. The corresponding `SINGULAR` commands are `opposite` (for defining the opposite algebra) and `oppose` (for mapping objects to the opposite algebra).

Further, `PLURAL` provides a command `twostd` which computes a Gröbner basis for the two-sided submodule generated by $f_1, \ldots, f_r$. The underlying algorithm proceeds along the following lines:

Step 0. Set $\mathcal{G} = \{f_1, \ldots, f_r\}$.
Step 1. Using Buchberger's algorithm, extend $\mathcal{G}$ to a left Gröbner basis for the left module generated by $\mathcal{G}$.
Step 2. For each $g \in \mathcal{G}$ and $i = 1, \ldots, n$, compute a remainder on left division of $g \cdot x_i$ by the Gröbner basis elements. If the remainder is nonzero, add it to $\mathcal{G}$ and go to Step 1.
Step 3. Return $\mathcal{G}$.

Example 3.45. Continuing the `SINGULAR` session from Example 3.41 (1), we compute the reduced left Gröbner basis `LSI` for the left ideal $I \subset D_3$ generated by the elements $f_1 = x_1^2\partial_2^2 + x_2^2\partial_3^2$ and $f_2 = x_1\partial_2 + x_3$ (with respect to $>_{\mathtt{dp}}$):

[8] At this writing, custom-built, fast implementations specializing on particular GR-algebras (such as the Weyl algebra or the exterior algebra) are still missing.

```
> ideal I = x(1)^2*d(2)^2+x(2)^2*d(3)^2, x(1)*d(2)+x(3);
> option(redSB);
> ideal LSI = std(I);
> LSI;
LSI[1]=x(1)*d(2)+x(3)
LSI[2]=x(3)^2
LSI[3]=x(2)*x(3)-x(1)
LSI[4]=x(1)*x(3)
LSI[5]=x(2)^2
LSI[6]=x(1)*x(2)
LSI[7]=x(1)^2
```

Next, we compute the right Gröbner basis RSI for the right ideal generated by f_1 and f_2 and check whether RSI is contained in I (by computing left normal forms for the elements of RSI mod I):

```
> def D3_opp = opposite(D3);
> setring D3_opp;   // active ring is the opposite algebra of D3
> basering;
//   characteristic : 0
//   number of vars : 6
//        block   1 : ordering a
//                  : names    D(3) D(2) D(1) X(3) X(2) X(1)
//                  : weights     1    1    1    1    1    1
//        block   2 : ordering ls
//                  : names    D(3) D(2) D(1) X(3) X(2) X(1)
//        block   3 : ordering C
//   noncommutative relations:
//    X(3)D(3)=D(3)*X(3)+1
//    X(2)D(2)=D(2)*X(2)+1
//    X(1)D(1)=D(1)*X(1)+1
> ideal I = oppose(D3,I);   // map I to opposite algebra
> ideal RSI_opp = std(I);
> setring D3;
> ideal RSI = oppose(D3_opp,RSI_opp);
> RSI;
RSI[1]=x(1)*d(2)+x(3)
RSI[2]=x(3)^2
RSI[3]=x(2)*x(3)+x(1)
RSI[4]=x(1)*x(3)
RSI[5]=x(2)^2
RSI[6]=x(1)*x(2)
RSI[7]=x(1)^2
> size(reduce(RSI,LSI));
1
```

The output 1 indicates that the right ideal generated by f_1, f_2 is not contained in I. This is not a surprise since the Weyl algebra over a field of characteristic zero is **simple**, that is, there is no nontrivial two-sided ideal in D_n (see, for instance, McConnel and Robson (2001), Section 1.3.1). In our example, we compute:

```
> ideal SI = twostd(I);
> SI;
SI[1]=1
```

□

Remark 3.46 (Elimination of Variables). Let $R = K\langle \boldsymbol{x}\rangle/J_0$ be a G-algebra (with notations as in Definition 3.37). Suppose that, for some $1 \leq k < n$ and all $k < i < j \leq n$, the letters $x_1, \ldots, x_k$ do not occur in h_{ij}. Then the two-sided ideal $J_k := J_0 \cap K\langle x_{k+1}, \ldots, x_n\rangle \subset K\langle x_{k+1}, \ldots, x_n\rangle$ is generated by the elements $x_j x_i - c_{ij} x_i x_j - h_{ij}$, $k < i < j \leq n$. If $I \subset R$ is a left ideal, we may aim at computing the ***k*th elimination ideal**

$$I_k := I \cap K\langle x_{k+1}, \ldots, x_n\rangle / J_k\,.$$

In analogy to Proposition 2.30 in Lecture 2, this problem is solvable if there is a global monomial order on $K[\boldsymbol{x}]$ which has the elimination property with respect to $x_1, \ldots, x_k$ *and* which is admissible for R.

However, such an order does not always exist. For instance, consider the G-algebra $K\langle x_1, x_2\rangle/\langle x_2x_1 - x_1x_2 - x_1^2\rangle$. If $>$ is an admissible order for R, then $x_1x_2 > x_1^2$ and, thus, $x_2 > x_1$. So $>$ does not have the elimination property with respect to x_1. □

The related **elimination problem for module components** is always solvable over R. In fact, it can be settled as in the commutative case (see Section 3.6.2): if $>$ is an admissible monomial order for R, and if F is the free R-module R^s with its canonical basis $e_1, \ldots, e_s$, then $>_{(\texttt{c},>)}$ is admissible for F (and an elimination order with respect to $e_1, \ldots, e_k$ for each k).

As an application of this, left syzygies and free resolutions over GR-algebras can be computed using the alternative method introduced in Section 3.6.6. In SINGULAR, use the syz, mres and nres commands (at this writing, Schreyer's algorithm is not yet implemented for the noncommutative case).

In theoretical terms, Schreyer's algorithm allows one to show that Hilbert's syzygy theorem holds for G-algebras. Over arbitrary GR-algebras, however, a given module may not have a finite free resolution:

Example 3.47. We continue our SINGULAR session from Example 3.41 (2). Considering the coefficient field $\mathbb{Q}$ as the graded E_3-module $\mathbb{Q} = E_3/\langle x_1, x_2, x_3\rangle$ which consists of one graded piece sitting in degree 0, we compute its minimal free resolution:

```
> ideal I = maxideal(1);
> def rI = mres(I,0);
// ** full resolution in a qring may be infinite,
//    setting max length to 5
> print(betti(rI),"betti");
```

```
           0     1     2     3     4     5
------------------------------------------
    0:     1     3     6    10    15    21
------------------------------------------
total:     1     3     6    10    15    21
> print(rI[1],"");
x(3),x(2),x(1)
> print(rI[2]);
x(3),x(2),0,   x(1),0,   0,
0,   x(3),x(2),0,   x(1),0,
0,   0,   0,   x(3),x(2),x(1)
```

□

Remark 3.48. To get more information on PLURAL and the commands provided for computations over noncommutative GR-algebras, type

```
> help plural;
```

For further details, see Levandovskyy (2005). □

3.8 Writing SINGULAR Procedures and Libraries

In SINGULAR, the user may enlarge the set of commands available by adding his own procedures. These are either written in the SINGULAR user language or in C/C++. In these notes, we only address the first type of procedures, referring to them as SINGULAR procedures (see the footnote on page 112 for a reference on how to deal with procedures written in C/C++). The general structure of a SINGULAR procedure is as follows:

```
proc ⟨procedure name⟩ (⟨parameter list⟩)
"⟨help text of the procedure⟩"
{
    ⟨procedure body⟩
}
```

The help text is optional but highly recommended if the procedure is meant to be stored in a file for later use. For instance:

```
proc sum(int n,m)
"USAGE:   sum(n,m), n,m int.
RETURN:   int, the sum n+(n+1)+...+m
"
{
   int i,N;
   for (i=n; i<=m; i++)
   {
      N=N+i;
   }
   return(N);
}
```

If this procedure has been written to the file, say, `sum.sing`, it can be read into a SINGULAR session and executed as follows:

```
> <"sum.sing";
> sum(3,7);
25
```

A SINGULAR library is a text file collecting several SINGULAR procedures. The recommended format for a library is as follows:

```
/**** header of the library ****/
version = "⟨version of the library⟩";
info = "⟨general help text for the library⟩";

/**** other libraries to be loaded ****/
LIB "⟨library 1⟩"; ..... LIB "⟨library n⟩";


/****** procedure ******/
proc ⟨procedure name⟩ (⟨parameter list⟩)
"⟨help text of the procedure⟩"
{
      ⟨procedure body⟩
}
example {
      ⟨example of usage⟩
}


/**** next procedure ****/
...
```

Again, the help texts and the example sections are optional, at least as long as the library is not to be included in the official SINGULAR package. Recall that a library can be loaded into a SINGULAR session using the LIB command.[9]

Our experience is that the most efficient way of writing a new library is to use one of the official SINGULAR libraries, say `algebra.lib`, as a sample. On a Unix-like operating system, type

```
> LIB "algebra.lib";
// ** loaded [...]
```

to see where the libraries are stored on your disk.

[9] A collection of procedures written in C/C++ can be linked to SINGULAR as a **dynamic module** during a SINGULAR session. At this writing, this feature is only provided for Unix-like operating systems. See Frühbis-Krüger, Krüger, and Schönemann (2003).

Example 3.49 (Debugging Tools). SINGULAR offers several tools which may help to debug a procedure or library:

`~`	Break point inside a procedure.
`listvar();`	Display all objects directly accessible from the active ring.
`listvar(⟨type-name⟩);`	Display all objects of type ⟨`type-name`⟩ directly accessible from the active ring.
`listvar(⟨ring-name⟩);`	Display all objects belonging to the ring ⟨`ring-name`⟩.
`printlevel`	Integer variable used to control the output of `dbprint`.
`dbprint(⟨level⟩,⟨text⟩);`	⟨`text`⟩ is printed iff the integer ⟨`level`⟩ is strictly positive.
`TRACE`	Integer variable used to set debugging level.

We discuss some of these tools by considering a short procedure which, given an integer n, computes $n!$ and prints the result on the screen. We define the procedure such that it depends on a parameter `n` of type `int`. Recall, however, that the range of integers of type `int` is rather limited (see Remark 3.7). Hence, already for relatively small values of n, the value of $n!$ exceeds this range. To allow a larger range for the computation, we make the procedure work with elements of the coefficient field of a ring of characteristic zero which virtually have no limitation (the corresponding data type is `number`). Since the active ring when calling the procedure may not be as desired, we define an auxiliary ring in the body of the procedure:

```
proc fac(int n)
"USAGE:   fac(n), n int.
RETURN:   None
NOTE:     displays n!
"
{
   ring S = 0, x, dp;
   number N = 1;
   int l;
   for (l=2; l<=n; l++)
   {
      N = N*l;
      dbprint(printlevel-voice+2,
               "// "+string(l)+"! successfully computed");
   }
   print(N);
}
```

Executing our procedure, the `string` given as a second argument of `dbprint` will be displayed iff the value of `printlevel-voice+2` is strictly positive.

Notice that `printlevel` is a predefined variable whose value can be changed by the user, while `voice` is an internal variable, representing the nesting level of procedures. In this way, we can control the nesting level up to which the `string` will be displayed if our procedure is called by other procedures. Having entered the procedure in a SINGULAR session, we apply it as follows:

```
> fac(5);
120
```

To get the additional output provided by the `dbprint` command, we have to raise the value of `printlevel`:

```
> printlevel;
0
> voice-1;        // display current nesting level
0
> printlevel = 1;
> fac(5);
// 2! successfully computed
// 3! successfully computed
// 4! successfully computed
// 5! successfully computed
120
```

Using the `listvar()` command to make SINGULAR display all objects which are directly accessible, we see, in particular, that there is not an active ring yet:

```
> listvar();
```

Indeed, the objects defined in our procedure are **local** in the sense that they only exist while the procedure is being executed, and that they differ from objects of the same name defined elsewhere.

In writing the procedure `fac`, we could have inserted the symbol ~ to mark a break point. For instance, we could have replaced the last three lines of the procedure by the lines below:

```
   }
   ~
   print(N);
}
```

If the break point is reached while executing the procedure, SINGULAR waits for input (without displaying any prompt).

```
> fac(5);
// 2! successfully computed
// 3! successfully computed
// 4! successfully computed
// 5! successfully computed

-- break point in ::fac --
-- called from STDIN --
```

We may now access objects which exist only during the execution of the procedure. To see what is available, enter again `listvar();`:

```
listvar();
// l                    [1]  int 6
// S                    [1]  *ring
//      N                    [1]  number
// n                    [1]  int 5

-- break point in ::fac --
```

The numbers in square brackets refer to the nesting level in which the objects are accessible (see also the footnote on Page 119). Typing `voice-1;`, we see that the current nesting level is 1. Hence, all objects listed above are accessible. For instance, typing `N;`, we get the result `120`.

Having reached a break point, we may also define new local objects, assign new values to given objects, or turn local objects into global ones by applying the `export` command (see Remark 3.50). For instance, entering

```
int k = int(N);
export(k);
```

makes `SINGULAR` create a new (local) variable `k` of type `int`, assign the computed value $5! = 120$ to `k` (type conversion), and declare `k` to be a global object (of the name space `Top`, see Remark 3.51).

Pressing the `enter` button on an empty line of input, the execution of the procedure is continued and the result is printed. We may now check that the variable `k` is still accessible:

```
> listvar();
// k                    [0]  int 120
```
□

Remark 3.50 (The Commands `return` and `export`). The commands `return` and `export` are used in the body of a procedure to make local data accessible elsewhere. The first command makes the procedure return the value and type of a local object (or, several local objects), terminating the execution of the procedure at the same time. The value is then accessible for later use if the procedure call assigns it to an object of the appropriate type. The second command exports the local object as a whole, making it a global object which is accessible under its original name (see Example 3.52 for a more precise statement in terms of name spaces).

Note that applying these commands to ring dependent objects has to be done with some care. If the active ring when executing the `return` or `export` command inside the procedure is not the ring which was active when calling the procedure, the returned or exported ring dependent data will not be accessible (`return` brings forth an error message). To overcome this problem, we have to implement the procedure in such a way that it exports the desired ring dependent object *and* returns its ring for subsequent use. For instance, in our procedure `fac`, we could have replaced the last three lines by the lines below:

```
    }
    export(N);
    return(S);
}
```

Then `fac` has a return value of type `ring`. Together with this ring comes an object `N` of type `number` (the result of our computation). This object may be accessed as follows:

```
> def RR = fac(5);
> setring RR;
> N;
120
```

Warning. When `export`ing a local object, SINGULAR does not check whether there is already a global object with the same name. This may cause different kinds of conflicts. Hence, we recommend to use the `export` command only if it is really needed (that is, the `return` command cannot be used instead). □

Remark 3.51 (Name Spaces). To assist the user in avoiding name conflicts, SINGULAR makes use of the concept of **name spaces**. Roughly speaking, this means that the system groups all objects defined in a SINGULAR session into several collections (the name spaces) so that names of objects in different collections cannot interfere with each other. The SINGULAR data type for name spaces is `package`. Entering `listvar(package);`, all name spaces defined in the current session are displayed. Having just started the session, we get:

```
> listvar(package);
// Standard                [0]  package (S,standard.lib)
// Top                     [0]  package (N)
```

At each stage of a SINGULAR session, precisely one of the name spaces is **active** in the following sense:

- Objects created at that stage (except those of type `package`) are assigned to the active name space (unless another name space is specified in the definition of the object).
- Objects assigned to the active name space are accessible under their name, while for objects from other name spaces the prefix ⟨`package-name`⟩`::` has to be added.[10]

[10] An object assigned to `Top` is accessible without prepending `Top::` unless an object of the same name is assigned to the active name space.

We illustrate this by continuing our SINGULAR session above:

```
> int i = 1;            // assign i to the active name space Top
> int Standard::j = 1;  // assign j to the name space Standard
> i;
1
> j;                    // as Standard is not active, j is not found
? `j` is undefined
? error occurred in STDIN line 5: `j;`
> Standard::j;          // prepending Standard::, j is accessible
1
```

Note that Top is the active name space whenever SINGULAR offers one of the prompts . or >.

For each library loaded into a SINGULAR session, the system creates an object of type package associated to that library.[11] The name of the package is obtained from the name of the library by capitalizing the first letter and removing the suffix .lib. For instance:

```
> LIB "grwalk.lib";
> listvar(package);
// Grwalk                  [0]  package (S,grwalk.lib)
// Standard                [0]  package (S,standard.lib)
// Top                     [0]  package (N)
```

The name space associated to a library is active precisely when a procedure of the library is being executed.[12]

A large library may contain many auxiliary procedures which are meant for internal use only. To prevent the names of these procedures from interfering with the names of user defined objects or procedures from other libraries, the auxiliary procedures can be declared to be static. In this case, the procedure is not user accessible in a SINGULAR session. □

We illustrate the concepts of name spaces and static procedures in the following example.

Example 3.52. Consider the three SINGULAR procedures below :

```
static proc add_one(int n)
{
  return(n+1);
}

proc namespaceDemo(int n)
{
```

[11] The same applies to dynamic modules.

[12] In these notes, we do not encounter any other situation in which the active name space is different from Top.

```
  n = add_one(n);
  def R = namespaceDemo1(n);
  return(R);
}

proc namespaceDemo1(int n)
{
  int k,m = 1,-1;
  n = add_one(n);
  export m;         // turn m into a global object
  ring R; poly f,g;
  export f;         // turn f into a global object (of R)
  ~                 // break point
  return(R);
}
```

We collect the procedures in a library named xxxx.lib and begin a SINGULAR session by loading this library:[13]

```
> LIB "xxxx.lib";
> listvar(package);
// Xxxx                   [0]  package (S,xxxx.lib)
// Standard               [0]  package (S,standard.lib)
// Top                    [0]  package (N)
```

We check that the static procedure add_one is not accessible for us:

```
> add_one(3);
   ? `add_one(3)` is undefined
   ? error occurred in STDIN line 3: `add_one(3);`
```

Next, we define two objects n, m of type int (which are automatically assigned to the active name space Top). Then, we apply the procedure namespaceDemo to n, assigning the return value to a variable named R. The execution of the procedure namespaceDemo1, which is called by namespaceDemo, is interrupted at the break point:

```
> int m,n = 2,3;
> def R = namespaceDemo(n);

-- break point in xxxx.lib::namespaceDemo1 --
-- called from xxxx.lib::namespaceDemo --
-- called from STDIN --
listvar();
// R                      [2]  *ring
```

[13] Instead of LIB, we could also use the load command for this. Using either of the commands, all library procedures will be assigned to the name space Xxxx. If LIB is used, the procedures which are not declared to be static are in addition considered to be assigned to Top.

```
//      g                    [2]  poly
//      f                    [0]  poly
// m                    [0]  int -1
// k                    [2]  int 1
// n                    [2]  int 5
// R                    [1]  def
// n                    [1]  int 4

-- break point in xxxx.lib::namespaceDemo1 --
m; n;
-1
5

-- break point in xxxx.lib::namespaceDemo1 --
```

Note that `listvar()` displays only objects assigned to the active name space `Xxxx`. We see that the names `m`, `n` refer to objects of `Xxxx`.[14] The objects with the same names `m`, `n` assigned to the name space `Top` are accessible as follows:

```
Top::m; Top::n;
2
3
-- break point in xxxx.lib::namespaceDemo1 --
```

Pressing the `enter` button one more time, the execution of the procedure is continued. After termination, `Top` is the active name space again:

```
> setring R;
> listvar();
// R                    [0]  *ring
//      f                    [0]  poly
// n                    [0]  int 3
// m                    [0]  int 2
> listvar(Xxxx);   // display all objects belonging to Xxxx
// Xxxx                 [0]  package (S,xxxx.lib)
// ::m                  [0]  int -1
// ::namespaceDemo1     [0]  proc from xxxx.lib
// ::namespaceDemo      [0]  proc from xxxx.lib
// ::add_one            [0]  proc from xxxx.lib (static)
```

We see that applying `export` to the ring independent local object `m` makes `m` a global object in the name space `Xxxx` (that is, the name space which is active while the procedure is being executed). On the other hand, applying `export` to a ring dependent local object `f` makes `f` a global object of the ring

[14] In searching for an object of a given name, SINGULAR begins in the current (nesting) level of the active name space. Then it turns to level 0 of the same name space, before it, finally, checks level 0 of `Top`. The search is terminated as soon as an object of the desired name has been found. If such object does not exist, an error message will be printed.

R to which it belongs. To make f accessible from Top (that is, the name space which was active when calling the procedure), the ring R itself has to be either returned or exported. □

Remark 3.53 (Modifying the Active Ring). It is occasionally necessary to design a procedure such that during its execution, the active ring is modified (for instance, we may wish to change the monomial order or to add a slack variable). In SINGULAR versions prior to 3-0-0, this often required the use of the execute command (see the solution to Exercise 5.1). Now, we strongly recommend to use the ringlist command which is custom-made for what we have in mind. If applied to the active ring, the ringlist command returns a list of data which determine the ring. We demonstrate this in the following SINGULAR session:

```
> ring R = (0,a), (x(1..3),y(1..2),z(1..2)), (dp(3),wp(2,5),lp);
> minpoly = a^2+1;
> qring Q = std(y(1)^2-x(1));
> list L = ringlist(Q);
> size(L);
4
```

So the list L has four entries. The first entry, which describes the coefficient field, is a list with four entries itself:

```
> L[1];
[1]:
   0
[2]:
   [1]:
      a
[3]:
   [1]:
      [1]:
         lp
      [2]:
         1
[4]:
   _[1]=(a^2+1)
```

We see that L[1][1], L[1][2], and L[1][4] are meant to store the characteristic (of type int), the parameters, and the minimal polynomial of the coefficient field. At this writing, the entry L[1][3], which refers to a monomial order, is mainly meant for internal use (giving the list L[1] a structure similar to that of L).

The second entry L[2] is a list of strings such that L[2][i] is the name of the ith variable. For instance:

```
> L[2][7];
z(2)
```

The entry `L[3]` is a list of `lists` describing the blocks of the monomial order (as displayed by the `basering` command, see Page 70). In this way, the monomial order on the ring and its extension to free modules are specified:

```
> L[3];
[1]:
   [1]:
      dp
   [2]:
      1,1,1
[2]:
   [1]:
      wp
   [2]:
      2,5
[3]:
   [1]:
      lp
   [2]:
      1,1
[4]:
   [1]:
      C
   [2]:
      0
```

The first entry of `L[3][i]` is always of type `string`, referring to the name of a monomial order. The second entry provides further information on the monomial order and indicates the size of the block. In our example, this is done by entries of type `intvec`, specifying the weights of the variables (and the size of the block). See the online help for matrix orders.

Finally, the fourth entry of `L` is the ideal defining the quotient ring (this ideal is zero if the ring under consideration is of type `ring`).

```
> L[4];
_[1]=x(1)-y(1)^2
```

Given a list `L` as above, we can modify it and define a new ring, say `S`, by applying the `ring` command. We illustrate this by changing the name of the parameter, adding an extra variable `w` (and changing the monomial order accordingly):

```
> L[1][2][1] = "b";              // new name for the parameter
> L[2][8] = "w";                 // append a new variable with name w
> L[3][3][2] = intvec(1,1,1);    // raise the size of the third block
>                                // of the monomial order
. def S = ring(L);
> setring S;
> basering;
//   characteristic : 0
//   1 parameter    : b
//   minpoly        : (b^2+1)
//   number of vars : 8
//        block   1 : ordering dp
//                  : names    x(1) x(2) x(3)
//        block   2 : ordering wp
//                  : names    y(1) y(2)
//                  : weights     2    5
//        block   3 : ordering lp
//                  : names    z(1) z(2) w
//        block   4 : ordering C
// quotient ring from ideal
_[1]=x(1)-y(1)^2
```

□

3.9 Communication with Other Systems

SINGULAR can be used as a support for other computer algebra systems. What we present here is the most basic way of communicating between MAPLE and SINGULAR on Unix-like platforms, writing to and reading from files (a similar way of communication should be possible for almost all computer algebra systems in place of MAPLE). More advanced scripts for data exchange are given in Greuel and Pfister (2002), Appendix B.

Example 3.54 (SINGULAR Support for a MAPLE Session). Given polynomials in a MAPLE session, we use SINGULAR to compute a lexicographic Gröbner basis for the ideal generated by the polynomials and read the result into the MAPLE session. To begin with, we make MAPLE write the polynomials to a file, say SINGULAR_in, using a format which SINGULAR can understand. Here is the corresponding MAPLE code:

```
> f1 :=  x^7+y^7:
> f2 :=  y^7+z^7:
> f3 :=  x^7+z^7+2:
> f4 :=  x^6*y+y^6*z+z^6+x:
> interface(prettyprint=0);
> interface(echo=0);
> writeto(SINGULAR_in);
lprint(‘ideal I = ‘);
f1, f2, f3, f4;
lprint(‘;‘);
writeto(terminal);
>
```

The resulting file looks as follows:

```
ideal I =
x^7+y^7, y^7+z^7, x^7+z^7+2, x^6*y+y^6*z+z^6+x
;
```

Now, we run a SINGULAR session:

```
> ring R = 0, (x,y,z), lp;
> option(redSB);
> <"SINGULAR_in";
> short = 0;              // enforce long format for the output
> ideal J = groebner(I);
> write(":w MAPLE_in","SingResult:=[");
> write(":a MAPLE_in",J);
> write(":a MAPLE_in","];");
```

The last three lines make SINGULAR write the Gröbner basis J to a file named MAPLE_in, using a format which MAPLE can understand. This file looks as follows:

```
SingResult:=[
z^7+1,y*z^6-y*z^5+y*z^4-y*z^3+y*z^2-y*z+y-z^6+z^5-z^4+z^3-z^2+z-1,y^2
-y*z^2-y+z^2,x+y*z^5-y*z^4+y*z^3-y*z^2+y*z-y+z^6-z^5+z^4-z^3+z^2-z+1
];
```

The file MAPLE_in can be read into the MAPLE session by typing

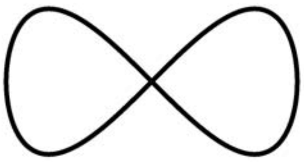
```
> read(MAPLE_in):
> SingResult;
[z^7+1, y*z^6-y*z^5+y*z^4-y*z^3+y*z^2-y*z+y-z^6+z^5-z^4+z^3-z^2+z-1,
y^2-y*z^2-y+z^2, x+y*z^5-y*z^4+y*z^3-y*z^2+y*z-y+z^6-z^5+z^4-z^3+z^2
-z+1]
```

□

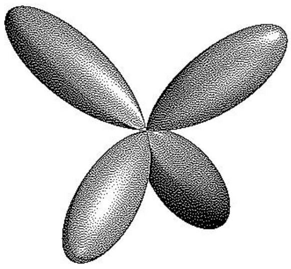
3.10 Visualization: Plotting Curves and Surfaces

If the software SURF is correctly installed on your computer, you may call it from inside a SINGULAR session to plot a picture of a plane curve, respectively of a surface in 3-space:

```
> LIB "surf.lib";
> ring r = 0, (x,y), dp;
> poly f = x2*(1-x2)-y2;
> plot(f,"scale_x=0.11; scale_y=0.11;");
```



```
> ring R = 0, (x,y,z), dp;
> plot(x4+y4+z4-15*xyz);
```



Remark 3.55 (Further Reading). For more information on SINGULAR and for proofs of the results presented in Section 3.6, see Greuel and Pfister (2002), Chapters 1, 2, and Appendix B. For more details on noncommutative Gröbner bases, we refer to Mora (1986), Apel (1988), Mora (1994), and Li (2002).

Practical Session I

Exercise 1.1. Start a `SINGULAR` session.

(a) Define a ring by typing `ring R;`. Then type `R;` or `basering;` to obtain information on the ring R. Observe that R is the polynomial ring $\mathbb{F}_{32003}[x, y, z]$ equipped with the degree reverse lexicographic order `dp`. Define the polynomial $f = x^4 + x^3z + x^2y^2 + yz^4 + z^5$. Print f by typing `f;`. How are the monomials of f ordered?

(b) Use the `ring` command to define a new ring S which differs from R only by the choice of the monomial order: choose the lexicographic order `lp`. Use the command `fetch` to map f from R to S (call the "new" polynomial g). Print g. How are the monomials ordered now?

Exercise 1.2. Generate 10 homogeneous random polynomials in 5 variables of degree 5 over a **finite (!)** field (of your choice).

(a) Compute a lexicographic Gröbner basis for the ideal generated by the 10 polynomials (use `timer` to check the computing time).
(b) How many Gröbner basis elements do you get? Print all elements.
(c) Print the degree of the first and the last element, respectively.
(d) Write the computed Gröbner basis to a file named `lexGB.out` (use the `write` command).
(e) Repeat (a) – (c), replacing the lexicographic order `lp` by the degree reverse lexicographic order `dp`. Write the computed Gröbner basis to a file named `dpGB.out`.

Exercise 1.3. Define the matrix

$$M = \begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ x_1 & x_2 & x_3 & x_4 \end{pmatrix}$$

and the ideal $I \subset \mathbb{Q}[x_0, \ldots, x_4]$ generated by its 2×2 minors.

(a) Compute the minimal free resolution of I and print the corresponding Betti diagram.
(b) Print the syzygy matrices. What is their data type ?
(c) Compute the Hilbert series of $\mathbb{Q}[x_0, \ldots, x_4]/I$.
(d) Check that $\mathrm{V}(I) \subset \mathbb{P}^4(\mathbb{C})$ is smooth.

Exercise 1.4. This exercise is concerned with `SINGULAR` procedures. Check the `SINGULAR` on-line help system for `proc`.

(a) Write a `SINGULAR` procedure which takes as input an `ideal` I, given by generators $f_1, \dots, f_r$, and which returns the maximum degree of the f_i.
(b) Apply the procedure to the Gröbner bases computed in Exercise 1.2 (`read` the data from the files `lexGB.out`, respectively `dpGB.out`).

Exercise 1.5. Let `VP5` denote the **Veronese surface in** $\mathbb{P}^5$, namely $\mathbb{P}^2$ embedded by the map

$$\mathbb{P}^2 \to \mathbb{P}^5\,,\ (u:v:w) \mapsto (u^2 : v^2 : w^2 : uv : uw : vw)\,,$$

where $\mathbb{P}^n = \mathbb{P}^n(\mathbb{C})$.

(a) Compute an ideal of $\mathbb{Q}[x_0, \dots, x_5]$ defining `VP5`. How many generators of which degree do you get? Check that `VP5` is smooth.
(b) Check that the point $p = (0:0:0:1:1:1)$ does not lie on `VP5`. The **Veronese surface `VP4` in** $\mathbb{P}^4$ is obtained by projecting `VP5` from p. Compute an ideal of $\mathbb{Q}[x_0, \dots, x_4]$ defining `VP4` (you should get seven cubic generators). Check that `VP4` is smooth.
(c) Randomly choose two cubics in the ideal defining `VP4`. Let `CI1` be the complete intersection defined by these two cubics. Then `CI1` is the union of `VP4` and some other surface, say `QES` (we say that `QES` is **linked** to `VP4` by `CI1`). Compute an ideal defining `QES` and its minimal free resolution. Print the Betti numbers. Check that `QES` is smooth.
(d) Check that the point $q = (1:1:1:1:1:1)$ lies on `VP5`. Compute an ideal defining the surface `CS` in $\mathbb{P}^4$ obtained by projecting `VP5` from q. Check that `CS` is smooth. Randomly choose two cubics in the ideal defining `CS`. Let `CI2` be the complete intersection defined by these two cubics. Compute an ideal defining the surface `B` which is linked to `CS` by `CI2`. How many generators of which degree do you get? Compute the minimal free resolution of the ideal defining `B` and print the Betti numbers. Check that `B` is smooth.
(e) Check that the line $L = \mathrm{V}(x_0+x_1+x_2, x_3, x_4, x_5)$ does not meet `VP5`. Compute a defining equation for the surface `SRS` obtained by projecting `VP5` from L to $\mathbb{P}^3$. You should get the homogenized equation of the Steiner Roman surface (see Example 2.33).

Remark. The constructions of the surfaces in this exercise depend on the choice of some parameters. For instance, instead of $u^2, v^2, w^2, uv, uw, vw$, we could take any other K-basis for the quadrics in u, v, w. Thus, we obtain a whole family of Veronese surfaces in $\mathbb{P}^5$, and, similarly, families of Veronese surfaces in $\mathbb{P}^4$ and families of surfaces of type `QES`, `CS`, and `B`. The latter surfaces are usually referred to as **quintic elliptic scrolls**, **cubic scrolls**, and **Bordiga surfaces**.

Practical Session II

Exercise 2.1. (a) Compute the algebra relations on the polynomials

$$f_1 = x^2 + y^2, \quad f_2 = x^2y^2, \quad f_3 = x^3y - xy^3 \in \mathbb{Q}[x,y].$$

(b) Consider the polynomials

$$g = x^4 + y^4, \quad g_1 = x + y, \quad g_2 = xy \in \mathbb{Q}[x,y].$$

Use `SINGULAR` to verify that g is contained in the subalgebra $\mathbb{Q}[g_1, g_2]$ of $\mathbb{Q}[x,y]$ and express g as a polynomial in g_1, g_2.

(c) Consider the endomorphism $\mathbb{Q}[x_1,x_2,x_3] \xrightarrow{\phi} \mathbb{Q}[x_1,x_2,x_3]$ defined by

$$x_1 \longmapsto x_2x_3\,, \quad x_2 \longmapsto x_1x_3\,, \quad x_3 \longmapsto x_1x_2\,.$$

Use `SINGULAR` to verify that ϕ induces an automorphism of

$$\mathbb{Q}[x_1,x_2,x_3]/\langle x_1x_2x_3 - 1\rangle.$$

Exercise 2.2. (a) The **affine twisted cubic curve** in $C \subset \mathbb{A}^2(\mathbb{C})$ is defined by the ideal

$$I = \langle f_1, f_2\rangle \subset \mathbb{Q}[x,y,z], \quad \text{where } f_1 = y - x^2, f_2 = z - x^3$$

(note that C is the image of the parametrization $t \mapsto \big(t, t^2, t^3\big)$). Compute the homogenization I^{hom} of I with respect to a further variable w, and compare the result with the ideal defining the projective twisted cubic curve in Example 1.18 of Lecture 1.

(b) Consider the ideal $J \subset \mathbb{Q}[w,x,y,z]$ generated by the homogenized polynomials $f_1^{\mathrm{hom}}, f_2^{\mathrm{hom}}$. By construction, J is contained in I^{hom}, so $\mathrm{V}(I^{\mathrm{hom}}) \subset \mathrm{V}(J) \subset \mathbb{P}^3(\mathbb{C})$. Verify this with `SINGULAR`. Further, verify that the inclusion of ideals is strict. In fact, $\mathrm{V}(J)$ has an "extra component at infinity". Check this by computing $\overline{\mathrm{V}(J) \setminus \mathrm{V}(I^{\mathrm{hom}})} \subset \mathbb{P}^3(\mathbb{C})$.

Exercise 2.3. A cubic curve in the projective plane $\mathbb{P}^2 = \mathbb{P}^2(\mathbb{C})$ is defined by a nonzero polynomial of type

$$F_{\boldsymbol{a}} = a_1x^3 + a_2x^2y + a_3xy^2 + a_4y^3 + a_5x^2z + a_6xyz + a_7y^2z + a_8xz^2 \\ + a_9yz^2 + a_{10}z^3\,,$$

where $\boldsymbol{a} = (a_1, \ldots, a_{10}) \in \mathbb{C}^{10}$. In the affine chart $U_z = \{z \neq 0\} \cong \mathbb{A}^2 = \mathbb{A}^2(\mathbb{C})$ of $\mathbb{P}^2$, it is defined by the dehomogenization $f_{\boldsymbol{a}}$ of $F_{\boldsymbol{a}}$ obtained from $F_{\boldsymbol{a}}$ by substituting 1 for z. Use elimination to compute defining equations for the Zariski closure of the locus of points $\boldsymbol{a} \in \mathbb{C}^{10}$ for which $\mathrm{V}(f_{\boldsymbol{a}})$ has a singular point p in the sense that $f_{\boldsymbol{a}}$ and all its first partial derivatives vanish at p. How many equations do you get? Are the equations homogeneous? What is their degree and how many terms do they have?

Exercise 2.4. Write `SINGULAR` procedures for computing

(a) ideal intersections,
(b) ideal quotients, and
(c) the saturation of an ideal.

Rely on syzygy computations as explained in Lecture 2 (see Page 52). Check all procedures by computing examples.

Lecture 4

Constructive Module Theory and Homological Algebra I

Homological algebra deals with constructions involving modules and complexes of modules. If R is a ring, a **complex** of R-modules is a sequence of R-modules and homomorphisms of R-modules

$$\mathcal{M}: \quad \cdots \longleftarrow M_{i-1} \xleftarrow{d_i} M_i \xleftarrow{d_{i+1}} M_{i+1} \longleftarrow \cdots$$

such that $d_i \circ d_{i+1} = 0$ for all i. The **homology** of $\mathcal{M}$ **at** $\boldsymbol{M_i}$ is defined to be $\ker d_i / \operatorname{im} d_{i+1}$. The sequence is **exact at** $\boldsymbol{M_i}$ if the homology at M_i is zero, and **exact** if it is exact at each M_i. Particular examples of complexes are free resolutions of modules as introduced in Lecture 1: if M is a (finitely generated) R-module, the "free part"

$$F_0 \xleftarrow{\varphi_1} F_1 \longleftarrow \cdots \longleftarrow F_{i-1} \xleftarrow{\varphi_i} F_i \xleftarrow{\varphi_{i+1}} F_{i+1} \longleftarrow \cdots$$

of a free resolution of M is exact at F_i for $i \geq 1$, and its homology at F_0 is M. Hilbert's proof of the polynomial nature of the Hilbert function via the syzygy theorem is an early application of homological algebra.

In this lecture, we discuss some of the basic constructions of homological algebra from a computational point of view.

4.1 Lifting Homomorphisms

Let $R = K[x_1, \ldots, x_n]$. We explain a construction which is central to many other constructions involving homomorphisms of R-modules.

Problem 4.1 (Lifting Homomorphisms). Given free R-modules F, G, H with fixed bases, and given homomorphisms $\varphi : H \to F$ and $\psi : G \to F$ such that

$$\operatorname{im} \psi \subset \operatorname{im} \varphi,$$

construct a "lift" $\widetilde{\psi} : G \to H$ such that $\varphi \circ \widetilde{\psi} = \psi$. That is, the diagram

$$\begin{tikzcd}G \arrow[d, "\psi"'] \arrow[dr, dashed, "\widetilde{\psi}"] & \\ F & H \arrow[l, "\varphi"]\end{tikzcd}$$

is commutative.

Solution. A lift $\widetilde{\psi}$ as desired exists iff $\operatorname{im}\psi \subset \operatorname{im}\varphi$. To explicitly construct it in this case, let $h_1, \dots, h_r$ be the images of the basis vectors of H under φ, and let $g_1, \dots, g_s$ be the images of the basis vectors of G under ψ. Since $\operatorname{im}\psi \subset \operatorname{im}\varphi$, and since the h_i generate $\operatorname{im}\varphi$, each g_j may be written as an R-linear combination $g_j = \sum_{i=1}^{r} a_{ij}h_i$ (follow the recipe given in the solution to the Submodule Membership Problem 2.16). The matrix (a_{ij}) gives $\widetilde{\psi}$. □

In SINGULAR, the command `lift` takes care of lifting homomorphisms. We refer to Section 4.2.4 for examples.

4.2 Constructive Module Theory

Let $R = K[x_1, \dots, x_n]$. We say that we have **constructed** a finitely generated R-module M if we can give finitely many generators and relations for it, that is, if we can describe M explicitly by means of an exact sequence

$$\begin{tikzcd}0 & M \arrow[l] & F_0 \arrow[l] & F_1 \arrow[l, "\varphi"']\end{tikzcd},$$

with free R-modules F_0, F_1. Recall from Lecture 1 that we refer to such a sequence as a **free presentation** of M. Since we can compute syzygies, every submodule of a free R-module which is explicitly given by finitely many generators may be regarded as having been constructed.

If M is given by means of a free presentation as above, and if N is another finitely generated R-module given by means of a free presentation, say

$$\begin{tikzcd}0 & N \arrow[l] & G_0 \arrow[l] & G_1 \arrow[l, "\psi"']\end{tikzcd},$$

then every homomorphism $\alpha : M \to N$ lifts to a **homomorphism of presentations**. That is, there is a commutative diagram

$$\begin{tikzcd}0 & M \arrow[l] \arrow[d, "\alpha"] & F_0 \arrow[l] \arrow[d, "\alpha_0"] & F_1 \arrow[l, "\varphi"'] \arrow[d, "\alpha_1"] \\ 0 & N \arrow[l] & G_0 \arrow[l] & G_1 \arrow[l, "\psi"']\end{tikzcd}.$$

Conversely, every pair $\alpha_0 : F_0 \to G_0$, $\alpha_1 : F_1 \to G_1$ of homomorphisms satisfying $\alpha_0 \circ \varphi = \psi \circ \alpha_1$ gives rise to a homomorphism $\alpha : M \to N$. If just a homomorphism $\alpha_0 : F_0 \to G_0$ is given, then α_0 descents to a homomorphism $\alpha : M \to N$ only if it can be lifted to a homomorphism $\alpha_1 : F_1 \to G_1$. That is, we need that α_0 takes the image of φ to the image of ψ.

4.2.1 Cokernels and Mapping Cones

If a homomorphism α between two finitely generated R-modules M and N as above is given, we have a commutative diagram with exact rows and column(s):

$$\begin{tikzcd}
0 & M \arrow[l] \arrow[d, "\alpha"] & F_0 \arrow[l] \arrow[d, "\alpha_0"] & F_1 \arrow[l, "\varphi"'] \arrow[d, dotted, "\alpha_1"] \\
0 & N \arrow[l] \arrow[d] & G_0 \arrow[l] & G_1 \arrow[l, "\psi"'] \\
 & \operatorname{coker}\alpha \arrow[d] & & \\
 & 0 & &
\end{tikzcd}$$

Chasing this diagram, we see that the induced sequence

$$0 \longleftarrow \operatorname{coker}\alpha \longleftarrow G_0 \xleftarrow{(\alpha_0,\psi)} F_0 \oplus G_1$$

is exact. It is, thus, a free presentation of $\operatorname{coker}\alpha$. So if α_0 and ψ are explicitly given as matrices, constructing $\operatorname{coker}\alpha$ just means to concatenate these matrices (in `SINGULAR`, use `concat` from `matrix.lib`).

If a **homomorphism of free resolutions** is given, that is, if we have a commutative diagram

$$\begin{tikzcd}
0 & M \arrow[l] \arrow[d, "\alpha"] & F_0 \arrow[l] \arrow[d, "\alpha_0"] & F_1 \arrow[l, "\varphi_1"'] \arrow[d, "\alpha_1"] & F_2 \arrow[l, "\varphi_2"'] \arrow[d, "\alpha_2"] & \cdots \arrow[l, "\varphi_3"'] \\
0 & N \arrow[l] & G_0 \arrow[l] & G_1 \arrow[l, "\psi_1"'] & G_2 \arrow[l, "\psi_2"'] & \cdots, \arrow[l, "\psi_3"']
\end{tikzcd}$$

we get a free resolution of $\operatorname{coker}\alpha$ by taking a **mapping cone**:

$$G_0 \xleftarrow{(\alpha_0,\psi_1)} F_0 \oplus G_1 \xleftarrow{\begin{pmatrix} -\varphi_1 & 0 \\ \alpha_1 & \psi_2 \end{pmatrix}} F_1 \oplus G_2 \xleftarrow{\begin{pmatrix} -\varphi_2 & 0 \\ \alpha_2 & \psi_3 \end{pmatrix}} F_2 \oplus G_3 \longleftarrow \cdots$$

4.2.2 Modulo

Many problems in constructive module theory can be reduced to solving systems of equations over R and, thus, to syzygy computations over R. As an example, we explain a construction which is central to many other constructions in homological algebra. The `SINGULAR` command for this construction is `modulo`.

Problem 4.2 (Modulo). Given free R-modules F, G, H with fixed bases, and given homomorphisms

$$\begin{tikzcd}
G \arrow[d, "\psi"'] & \\
F & H \arrow[l, "\varphi"'] ,
\end{tikzcd}$$

construct the R-module

$$(\operatorname{im}\varphi + \operatorname{im}\psi)/\operatorname{im}\psi.$$

Solution. Let $h_1, \dots, h_r$ be the images of the basis vectors of H under φ, and let $g_1, \dots, g_s$ be the images of the basis vectors of G under ψ. Computing the syzygies on $h_1, \dots, h_r, g_1, \dots, g_s$, we get a free presentation of $\operatorname{im}\varphi + \operatorname{im}\psi$, say

$$\begin{tikzcd}
0 & \operatorname{im}\varphi + \operatorname{im}\psi \arrow[l] & H \oplus G \arrow[l, "{(\varphi,\psi)}"'] & H' \arrow[l, "\binom{\alpha}{\beta}"'] .
\end{tikzcd}$$

The induced sequence

$$\begin{tikzcd}
0 & (\operatorname{im}\varphi + \operatorname{im}\psi)/\operatorname{im}\psi \arrow[l] & H \arrow[l, "\overline{\varphi}"'] & H' \arrow[l, "\alpha"']
\end{tikzcd}$$

is a free presentation of $(\operatorname{im}\varphi + \operatorname{im}\psi)/\operatorname{im}\psi$. Indeed, to see exactness, chase the following commutative diagram with exact rows and columns:

$$\begin{tikzcd}
 & 0 \arrow[d] & 0 \arrow[d] & \\
0 & \operatorname{im}\psi \arrow[l] \arrow[d] & G \arrow[l, "\psi"'] \arrow[d] & \\
0 & \operatorname{im}\varphi + \operatorname{im}\psi \arrow[l] \arrow[d] & H \oplus G \arrow[l, "{(\varphi,\psi)}"'] \arrow[d] & H' \arrow[l, "\binom{\alpha}{\beta}"'] \arrow[d, equal] \\
0 & (\operatorname{im}\varphi + \operatorname{im}\psi)/\operatorname{im}\psi \arrow[l] \arrow[d] & H \arrow[l, "\overline{\varphi}"'] \arrow[d] & H' \arrow[l, "\alpha"'] \\
 & 0 & 0 &
\end{tikzcd}$$

□

4.2.3 Kernel, Hom, Ext, Tor, and more

We use `modulo` to give solutions to several problems asking for constructions which are fundamental to homological algebra. In formulating the problems and solutions, we suppose that M, N and L are finitely generated R-modules given by means of free presentations

$$\begin{tikzcd}
0 & M \arrow[l] & F_0 \arrow[l] & F_1 \arrow[l, "\varphi"'] , & 0 & N \arrow[l] & G_0 \arrow[l] & G_1 \arrow[l, "\psi"'] ,
\end{tikzcd}$$

$$\begin{tikzcd}
0 & L \arrow[l] & E_0 \arrow[l] & E_1 \arrow[l, "\eta"'] .
\end{tikzcd}$$

Problem 4.3 (Image and Kernel). Given a homomorphism $\alpha : M \to N$ by means of a homomorphism $\alpha_0 : F_0 \to G_0$ taking the image of φ to the image of ψ, construct $\operatorname{im}\alpha$ and $\ker\alpha$.

Solution. To obtain $\operatorname{im}\alpha$, we just need a single `modulo` computation. An additional `modulo` computation yields $\ker\alpha$. We illustrate these constructions by two commutative diagrams with exact rows and columns:

$$\begin{tikzcd}
G_1 & H_0 \arrow[l, dotted] \arrow[d, "\beta_0"] & \\
G_0 \arrow[u, leftarrow, "\psi"'] \arrow[d] & F_0 \arrow[l, "\alpha_0"'] \arrow[d, "\overline{\alpha}_0"] & \\
N \arrow[d] & A \arrow[l] \arrow[d] & 0 \arrow[l] \\
0 & 0 &
\end{tikzcd}$$

$$\begin{tikzcd}
G_1 \arrow[d, "\psi"'] & F_1 \arrow[l, dotted] \arrow[d, "\varphi"'] & H_1 \arrow[l, dotted] \arrow[d, "\kappa"'] & \\
G_0 \arrow[d] & F_0 \arrow[l, "\alpha_0"'] \arrow[d] & H_0 \arrow[l, "\beta_0"'] \arrow[d, "\overline{\beta}_0"] & \\
N \arrow[d] & M \arrow[l, "\alpha"'] \arrow[d] & B \arrow[l] \arrow[d] & 0 \arrow[l] \\
0 & 0 & 0 &
\end{tikzcd}$$

In the first diagram, $A = (\operatorname{im}\alpha_0 + \operatorname{im}\psi)/\operatorname{im}\psi$, and the second column is obtained via `modulo`. It is clear from the construction in Problem 4.2 that α_0 descends to a monomorphism which embeds A as $\operatorname{im}\alpha$ into N.

In the second diagram, $B = (\operatorname{im}\beta_0 + \operatorname{im}\varphi)/\operatorname{im}\varphi$, the third column is obtained via `modulo`, and β_0 descends to a monomorphism which embeds B as $\ker\alpha$ into M. □

Problem 4.4 (Homology). Construct the homology at M of the complex

$$N \xleftarrow{\alpha} M \xleftarrow{\alpha'} L\,,$$

supposing that α and α' are given by means of homomorphisms $\alpha_0 : F_0 \to G_0$ and $\alpha_0' : E_0 \to F_0$ taking the image of φ to the image of ψ, respectively the image of η to the image of φ, and satisfying $\operatorname{im}(\alpha_0 \circ \alpha_0') \subset \operatorname{im}\psi$.

Solution. Having computed β_0 as in the construction of $\operatorname{im}\alpha$ described above, computing the homology at M requires only one further `modulo` computation:

$$\ker\alpha/\operatorname{im}\alpha' = (\operatorname{im}\beta_0 + \operatorname{im}\alpha_0' + \operatorname{im}\varphi)/(\operatorname{im}\alpha_0' + \operatorname{im}\varphi)\,. \qquad \square$$

Problem 4.5 (Hom). Construct $\operatorname{Hom}_R(M, N)$.

Solution. Since $\operatorname{Hom}_R(M, N)$ is the kernel of the map

$$\operatorname{Hom}_R(F_1, N) \xleftarrow{\operatorname{Hom}_R(\varphi, N)} \operatorname{Hom}_R(F_0, N)\,,$$

it can be computed from the free presentations

$$0 \longleftarrow \operatorname{Hom}_R(F_i, N) \longleftarrow \operatorname{Hom}_R(F_i, G_0) \longleftarrow \operatorname{Hom}_R(F_i, G_1)$$

which are induced by the given free presentation of N, $i = 0, 1$. □

Next, we consider the Ext functors which measure the extent to which the functors $\mathrm{Hom}(M,-)$ and $\mathrm{Hom}(-,N)$ fail to be exact: if

$$0 \longleftarrow C \longleftarrow B \longleftarrow A \longleftarrow 0$$

is a short exact sequence of R-modules, there are **long exact sequences of Ext**,

$$\begin{tikzcd}
 & \mathrm{Hom}_R(M,C) \arrow[dl, rounded corners, to path={ -- ([xshift=-2ex]\tikztostart.west) |- ([yshift=2ex]\tikztotarget.north) -- (\tikztotarget)}, phantom] & \mathrm{Hom}_R(M,B) \arrow[l] & \mathrm{Hom}_R(M,A) \arrow[l] & 0 \arrow[l] \\
\cdots & \mathrm{Ext}^1_R(M,C) \arrow[l] & \mathrm{Ext}^1_R(M,B) \arrow[l] & \mathrm{Ext}^1_R(M,A) \arrow[l] \arrow[from=1-2, to=2-4, rounded corners]
\end{tikzcd}$$

and

$$\begin{tikzcd}
 & \mathrm{Hom}_R(A,N) & \mathrm{Hom}_R(B,N) \arrow[l] & \mathrm{Hom}_R(C,N) \arrow[l] & 0 \arrow[l] \\
\cdots & \mathrm{Ext}^1_R(A,N) \arrow[l] & \mathrm{Ext}^1_R(B,N) \arrow[l] & \mathrm{Ext}^1_R(C,N) \arrow[l] \arrow[from=1-2, to=2-4, rounded corners]
\end{tikzcd}$$

Problem 4.6 (Ext). Construct the modules $\mathrm{Ext}^i_R(M,N)$.

Solution. From the given free presentation of M, compute a free resolution of M:

$$0 \longleftarrow M \longleftarrow F_0 \longleftarrow F_1 \longleftarrow F_2 \longleftarrow \cdots$$

Then $\mathrm{Ext}^i_R(M,N)$ is the homology of the induced complex

$$\cdots \longleftarrow \mathrm{Hom}_R(F_2,N) \longleftarrow \mathrm{Hom}_R(F_1,N) \longleftarrow \mathrm{Hom}_R(F_0,N) \longleftarrow 0$$

at $\mathrm{Hom}_R(F_i,N)$. □

We turn from Hom to $\otimes$. A free presentation of $M \otimes_R N$ is induced by the given free presentations of M and N:

$$0 \longleftarrow M \otimes_R N \longleftarrow F_0 \otimes_R G_0 \longleftarrow (F_0 \otimes_R G_1) \oplus (F_1 \otimes_R G_0).$$

The Tor functors measure the extent to which $M \otimes_R -$ fails to be exact: if $0 \leftarrow C \leftarrow B \leftarrow A \leftarrow 0$ is a short exact sequence of R-modules, there is a **long exact sequence of Tor**,

$$\begin{tikzcd}
 & \mathrm{Tor}^R_1(M,C) & \mathrm{Tor}^R_1(M,B) \arrow[l] & \mathrm{Tor}^R_1(M,A) \arrow[l] & \cdots \arrow[l] \\
0 & M \otimes_R C \arrow[l] & M \otimes_R B \arrow[l] & M \otimes_R A \arrow[l] \arrow[from=1-2, to=2-4, rounded corners]
\end{tikzcd}$$

There is also a long exact sequence of Tor for $- \otimes_R N$. Indeed, the commutativity of the tensor product, $M \otimes_R N \cong N \otimes_R M$, induces natural isomorphisms $\mathrm{Tor}^R_i(M,N) \cong \mathrm{Tor}^R_i(N,M)$, $i \geq 0$.

Problem 4.7 (Tor). Construct the modules $\operatorname{Tor}_i^R(M,N)$.

Solution. From the given free presentation of N, compute a free resolution of N:

$$0 \longleftarrow N \longleftarrow G_0 \longleftarrow G_1 \longleftarrow G_2 \longleftarrow \cdots$$

Then $\operatorname{Tor}_i^R(M,N)$ is the homology of the induced complex

$$0 \longleftarrow M \otimes_R G_0 \longleftarrow M \otimes_R G_1 \longleftarrow M \otimes_R G_2 \longleftarrow \cdots$$

at $M \otimes_R G_i$. □

We refer to Eisenbud (1995) and Hilton and Stammbach (1971) for a detailed discussion of Ext and Tor.

Remark 4.8 (Graded Structures). Let M and N be finitely generated graded R-modules. If $\alpha : M \to N$ is a graded homomorphism of degree 0, then the kernel, the image, and the cokernel of α inherit a grading, too. The same is true for $M \otimes_R N$ and, thus, for the modules $\operatorname{Tor}_i^R(M,N)$, $i \geq 0$.

With respect to Hom, let $\operatorname{Hom}_R(M,N)_d$ be the K-vector space consisting of all elements of $\operatorname{Hom}_R(M,N)$ which are **graded of degree** d (that is, which send homogeneous elements of degree e to homogeneous elements of degree $e+d$). Then $\bigoplus_d \operatorname{Hom}_R(M,N)_d$ is a submodule of $\operatorname{Hom}_R(M,N)$. In our case, the two modules coincide due to the assumption that M is finitely generated. Thus, $\operatorname{Hom}_R(M,N)$ is graded, and the same holds for the modules $\operatorname{Ext}_R^i(M,N)$, $i \geq 0$. For instance, the dual module $R(d)^\vee := \operatorname{Hom}_R\big(R(d),R\big)$ is equal to $R(-d)$. □

The `SINGULAR` commands for the constructions in this section are `kernel`, `homology`, `Hom`, `Ext_R`, `Ext`, `tensorMod`, and `Tor`. The use of these and the other commands discussed in this lecture is not restricted to the polynomial ring R (see Remark 3.36 for free resolutions over other rings). For instance, the commands make sense and work over any quotient ring $Q = R/I$. Note, however, that modules over a quotient ring need not have a free resolution of finite length. Thus, the modules $\operatorname{Ext}_Q^i(M,N)$ and $\operatorname{Tor}_i^Q(M,N)$ may be nonzero for arbitrary large integers i.

Example 4.9. Let $Q = K[w,x,y,z]/I$ be the homogenous coordinate ring of the twisted cubic curve in $\mathbb{P}^3(K)$, and let $\langle w,x,y,z\rangle$ be the homogeneous maximal ideal of Q which is generated by the (residue classes of the) coordinates. Then

$$K = Q/\langle w,x,y,z\rangle$$

is a graded Q-module consisting of just one graded piece sitting in degree 0. We use `SINGULAR` to compute a part of its minimal free resolution. Note that for each i, the graded Betti number $\beta_{ii}(K)$ is the dimension of $\operatorname{Ext}_Q^i(K,K)$ as a K-vector space. All other graded Betti numbers are zero.

```
> ring R = 0, (w,x,y,z), dp;
> matrix m[2][3] = w,x,y,x,y,z;
> ideal I = minor(m,2);
> I;
I[1]=-y2+xz
I[2]=-xy+wz
I[3]=x2-wy
> qring Q = groebner(I);
> resolution F=mres(maxideal(1),0);
// ** full resolution in a qring may be infinite,
      setting max length to 6
> print(betti(F),"betti");
           0     1     2     3     4     5     6
------------------------------------------------
    0:     1     4     9    18    36    72   144
------------------------------------------------
total:     1     4     9    18    36    72   144
```

To obtain, say, 7 terms of the resolution, enter instead:

```
> resolution F = mres(maxideal(1),7);
> print(betti(F),"betti");
           0     1     2     3     4     5     6     7
------------------------------------------------------
    0:     1     4     9    18    36    72   144   288
------------------------------------------------------
total:     1     4     9    18    36    72   144   288
> print(F[1],"");
z,y,x,w
> print(F[2]);
y, 0, 0, x, 0, 0, w, 0, 0,
-z,y, 0, 0, x, 0, 0, w, 0,
0, -z,y, -z,0, x, 0, 0, w,
0, 0, -z,0, -z,-y,-z,-y,-x
```

Of course, the K-dimension of, say, $\operatorname{Ext}^7_Q(K,K)$ can also be computed via the `Ext` command:

```
> LIB "homolog.lib";
> module M = Ext(7,F[1],F[1]);
// dimension of Ext^7:  0
// vdim of Ext^7:       288
```
□

We refer to Avramov and Grayson (2002) for a method which, for finitely generated graded modules M, N over a graded complete intersection $Q = K[\boldsymbol{x}]/I$, computes all modules $\operatorname{Ext}^i_Q(M,N)$, $i \geq 0$, simultaneously. Here, we say that a quotient ring $K[\boldsymbol{x}]/I$ is a (graded) **complete intersection** if I is generated by a (homogeneous) regular sequence on $K[\boldsymbol{x}]$ (see Definition 5.24 for regular sequences).

4.2.4 Some Explicit Constructions

In this section, we explain how to construct graded homomorphisms between suitably twisted syzygy modules of the coefficient field K over the graded polynomial ring $K[x_0,\dots,x_n]$. The significance of these modules and of the graded homomorphisms between them comes from a theorem of Beilinson which has important applications in projective algebraic geometry. We will discuss the theorem of Beilinson in Appendix A.

We consider

$$K = K[x_0,\dots,x_n]/\langle x_0,\dots,x_n\rangle$$

as a graded $K[x_0,\dots,x_n]$-module consisting of one graded piece sitting in degree 0. In contrast to Example 4.9, the minimal free resolution of K over $S = K[x_0,\dots,x_n]$ must be finite due to Hilbert's syzygy theorem. In fact, its shape is as follows:

$$S \longleftarrow S(-1)^{\binom{n+1}{1}} \longleftarrow \dots \longleftarrow S(-n)^{\binom{n+1}{n}} \longleftarrow S(-n-1) \longleftarrow 0\,.$$

We will formally introduce this sequence in Lecture 5, referring to it as the **Koszul complex** defined by $x_0,\dots,x_n$ (recall that the minimal free resolution is determined up to an isomorphism of free resolutions). Twisting all modules of the Koszul complex by d, we get the minimal free resolution of $K(d)$, that is, of K, considered as a graded S-module sitting in degree $-d$.

Example 4.10. If $n = 2$, the Koszul complex reads

$$S \xleftarrow{(x_0,x_1,x_2)} S(-1)^3 \xleftarrow{\begin{pmatrix} x_1 & x_2 & 0 \\ -x_0 & 0 & x_2 \\ 0 & -x_0 & -x_1 \end{pmatrix}} S(-2)^3 \xleftarrow{\begin{pmatrix} x_2 \\ -x_1 \\ x_0 \end{pmatrix}} S(-3) \longleftarrow 0\,.$$

Dualizing it, we get the minimal free resolution of $K(3)$ (for arbitrary n, the twist is $n+1$). □

Example 4.11. We compute the Koszul complex in case $n = 3$:

```
> ring S = 0, x(0..3), dp;
> resolution kos = mres(maxideal(1),0);
> print(betti(kos),"betti");
           0     1     2     3     4
------------------------------------
    0:     1     4     6     4     1
------------------------------------
total:     1     4     6     4     1
```

Consider the graded syzygy modules $M = \mathrm{Syz}_3(K(2))$ and $N = \mathrm{Syz}_2(K(1))$. Our goal is to construct a nontrivial graded homomorphism $M \to N$ of degree zero.

To fix our ideas on how to proceed, we suppose for the moment that such a homomorphism α is given. Then α lifts to a homomorphism of free presentations

$$\begin{tikzcd}
0 & M \arrow[l] \arrow[d, "\alpha"'] & S(-1)^4 \arrow[l] \arrow[d, "\alpha_0"'] & S(-2) \arrow[l, "\varphi"'] \arrow[d, "\alpha_1"'] \\
0 & N \arrow[l] & S(-1)^6 \arrow[l] & S(-2)^4 \arrow[l, "\psi"']
\end{tikzcd}$$

such that the α_i are graded of degree 0. That is, the homomorphisms α_i are represented by matrices with constant entries. The composite map $\widetilde{\alpha}_1 := \psi \circ \alpha_1$ is represented by a matrix with linear entries.

Since $\psi \circ \alpha_1 = \alpha_0 \circ \varphi$, we have a commutative diagram of dual maps which is reminiscent of the Lifting Problem 4.1:

$$\begin{tikzcd}
S(1)^6 \arrow[d, "\widetilde{\alpha}_1^\vee"'] \arrow[dr, "\alpha_0^\vee"] & \\
S(2) & S(1)^4. \arrow[l, "\varphi^\vee"]
\end{tikzcd}$$

Reversing what we just did, we get a recipe for constructing the desired homomorphism. The crucial fact in the construction is that $\operatorname{im}\varphi^\vee$ is generated by $x_0, \ldots, x_3$. This implies that if α_1 is an arbitrarily chosen 4×1 matrix with constant entries, and if $\widetilde{\alpha}_1 := \psi \circ \alpha_1$, then $\operatorname{im}\widetilde{\alpha}_1^\vee \subset \operatorname{im}\varphi^\vee$. Hence, $\widetilde{\alpha}_1^\vee$ can be lifted to a homomorphism $\alpha_0^\vee$ which is represented by a 4×6 matrix with constant entries. The dual map α_0 descents to a graded homomorphism $\alpha : M \to N$ of degree 0. In our SINGULAR session, we choose α_1 at random, aiming at a "generic" α:

```
> matrix alpha1 = random(1000,4,1);  // randomly created intmat
> matrix tphi = transpose(kos[4]);
> matrix psi = kos[3];
> matrix talpha1tilde = transpose(psi*alpha1);
> matrix talpha0 = lift(tphi,talpha1tilde);
> print(talpha0);
35,834,102,0,  0,  0,
65,0,  0,  834,102,0,
0, 65, 0,  -35,0,  102,
0, 0,  65, 0,  -35,-834
```

□

In the next two examples, we construct homomorphisms whose cokernel is the ideal of a smooth surface in $\mathbb{P}^4 = \mathbb{P}^4(\mathbb{C})$. It is crucial to choose "generic" homomorphisms since otherwise the cokernel might be an ideal defining a singular surface. Even worse, it might not even be an ideal at all. The theory behind the constructions is best understood in the language of vector bundles and sheaves. We refer to Example A.9 in Appendix A for some explanations.

In fact, each homomorphism $M \to N$ to be constructed can be thought of as a homomorphism $\mathcal{F} \to \mathcal{G}$ of vector bundles on $\mathbb{P}^4$ (obtained by sheafifying $M \to N$). The ideal we are aiming at defines the degeneracy locus of this homomorphism. A variant of the Unmixedness Theorem 2.26 in Lecture 2, which applies to such a locus if the locus has the expected codimension, allows us to check smoothness using the Jacobian Criterion 2.23 (see Arbarello et al (1985), Chapter II for details). In the two examples below, $\operatorname{rank} \mathcal{G} = \operatorname{rank} \mathcal{F} + 1$. This means that the expected codimension is 2.

Especially in the second example, the computations based on the Jacobian criterion are expensive if performed over the integers (rationals).

Remark 4.12. As illustrated by Example 3.21 in Lecture 3, Gröbner basis computations in characteristic zero are quite involved due to the explosion of coefficients coming with Buchberger's algorithm. For experiments in algebraic geometry, it is therefore often advisable to perform the computations over a sufficiently large finite field, say over $K = \mathbb{F}_{32003}$. As experience shows, the geometric evidence resulting from such computations is usually identical to what we would get in characteristic zero. $\square$

To ease the smoothness check in the following examples, we choose $K = \mathbb{F}_{32003}$ as our coefficient field.

Example 4.13. Let $n = 4$. We compute the cokernel of a "generic" graded homomorphism $\alpha : M = S(-1)^3 \to N = \operatorname{Syz}_2(K(1))$ of degree 0:

$$\begin{tikzcd}
0 & S(-1)^3 \arrow[l] \arrow[d, "\alpha"'] & S(-1)^3 \arrow[l, "\mathrm{id}"'] \arrow[d, "\alpha_0"'] \arrow[dr, dash, "\oplus"] & 0 \arrow[l] & & \\
0 & \operatorname{Syz}_2(K(1)) \arrow[l] & S(-1)^{10} \arrow[l] & S(-2)^{10} \arrow[l] & S(-3)^5 \arrow[l] & S(-4) \arrow[l] & 0 \arrow[l]
\end{tikzcd}$$

On our way, we adjust the degrees using the `attrib` command such that the correct Betti numbers are displayed (see Lecture 3, Section 3.4 for the `attrib` command).

```
> ring S = 32003, x(0..4), dp;
> module MI=maxideal(1);
> attrib(MI,"isHomog",intvec(-1));
> resolution kos = nres(MI,0);
> print(betti(kos),"betti");
           0     1     2     3     4     5
------------------------------------------
   -1:     1     5    10    10     5     1
------------------------------------------
total:     1     5    10    10     5     1
> matrix alpha0 = random(32002,10,3);
> module pres = module(alpha0)+kos[3];
```

The 10×13 matrix `pres` is a presentation matrix for $\operatorname{coker} \alpha$. Resolving it, we get a free resolution and, thus, the graded Betti numbers of $\operatorname{coker} \alpha$:

```
> attrib(pres,"isHomog",intvec(1,1,1,1,1,1,1,1,1,1));
> resolution fcokernel = mres(pres,0);
> print(betti(fcokernel),"betti");
           0     1     2     3
------------------------------
    1:     7    10     5     1
------------------------------
total:     7    10     5     1
```

We see that, up to twist, the graded Betti numbers are equal to those of the ideal of a Veronese surface in $\mathbb{P}^4$ (which is minimally generated by seven cubics, see Exercise 1.5). Resolving "in the other direction", that is, resolving the transposed of `pres`, we indeed get a 7×1 matrix with cubic entries.

```
> module dir = transpose(pres);
> intvec w = -1,-1,-1,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2;
> attrib(dir,"isHomog",w);
> resolution fdir = mres(dir,2);
> print(betti(fdir),"betti");
           0     1     2
------------------------
   -2:    10     7     -
   -1:     -     -     -
    0:     -     -     1
------------------------
total:    10     7     1
```

We use the `SINGULAR` command `flatten` from `matrix.lib` which turns a matrix into the ideal generated by its entries.

```
> LIB "matrix.lib";
> ideal I = groebner(flatten(fdir[2]));
> resolution FI = mres(I,0);
> print(betti(FI),"betti");
           0     1     2     3     4
------------------------------------
    0:     1     -     -     -     -
    1:     -     -     -     -     -
    2:     -     7    10     5     1
------------------------------------
total:     1     7    10     5     1
```

Finally, we check that the ideal `I` generated by the seven cubics defines a smooth surface of degree 4 in $\mathbb{P}^4$. By classification results, this surface must be a Veronese surface (see, for instance, Okonek (1983)).

```
> int codimI = nvars(S)-dim(I);
> codimI;
2
```

```
> degree(I);
// dimension (proj.)  = 2
// degree (proj.)   = 4
> nvars(S)-dim(groebner(minor(jacob(I),codimI) + I));
5
```

□

Example 4.14. As in the preceeding example, let $n = 4$. Now, we compute the cokernel of a "generic" graded homomorphism

$$\alpha = (\beta, \gamma) : M = \mathrm{Syz}_4(K(3))^2 \longrightarrow N \oplus S = \mathrm{Syz}_2(K(1))^2 \oplus S$$

of degree 0. We begin by constructing $\beta : M \to N$ (following the recipe given in Example 4.11):

$$\begin{tikzcd}
0 & M \arrow[l] \arrow[d, "\beta"'] & S(-1)^{10} \arrow[l] \arrow[d, "\beta_0"'] & S(-2)^2 \arrow[l, "\varphi"'] \arrow[d, "\beta_1"'] \\
0 & N \arrow[l] & S(-1)^{20} \arrow[l] & S(-2)^{20} \arrow[l, "\psi"']
\end{tikzcd}$$

```
> LIB "matrix.lib";
> ring S = 32003, x(0..4), dp;
> resolution kos = nres(maxideal(1),0);
> betti(kos);
1,5,10,10,5,1
> matrix kos5 = kos[5];
> matrix tphi = transpose(dsum(kos5,kos5));
> matrix kos3 = kos[3];
> matrix psi = dsum(kos3,kos3);
> matrix beta1 = random(32002,20,2);
> matrix tbeta1tilde = transpose(psi*beta1);
> matrix tbeta0 = lift(tphi,tbeta1tilde);
```

Now, we turn to the construction of $\gamma : M \to S$. According to its definition, M fits as kernel into an exact sequence

$$0 \longleftarrow \mathrm{Syz}_3(K(3))^2 \longleftarrow S^{20} \longleftarrow M \longleftarrow 0.$$

Splitting the Koszul complex into short exact sequences and using the long exact sequence of Ext, we find that $\mathrm{Ext}^1_S\big(\mathrm{Syz}_3(K(3))^2, S\big)$ vanishes. This can also be seen using SINGULAR:

```
> LIB "homolog.lib";
> def E = Ext_R(1,kos[4]);
// dimension of Ext^1:  -1
```

Thus, it follows from the long exact sequence of Ext that every homomorphism $\gamma : M \to S$ lifts to a homomorphism $\gamma_{-1} : S^{20} \to S$ which fits into a commutative diagram

$$\begin{tikzcd}
\cdots & S^{20} \arrow[l] \arrow[d, "\gamma_{-1}"'] & S(-1)^{10} \arrow[l] \arrow[d, "\gamma_0"'] & \cdots \arrow[l] \\
0 & S \arrow[l] \arrow[r, equal] & S & 0 \arrow[l]
\end{tikzcd},$$

where the top row is the direct sum of two copies of the Koszul complex resolving $K(3)$. Conversely, every homomorphism $\gamma_{-1} : S^{20} \to S$ gives rise to a homomorphism $\gamma_0 : S(-1)^{10} \to S$ which descends to a homomorphism $\gamma : M \to S$. If γ_{-1} is graded of degree zero, then so are γ_0 and γ. In SINGULAR, we construct such homomorphisms as follows:

```
> matrix kos4 = kos[4];
> matrix tkos4pluskos4 = transpose(dsum(kos4,kos4));
> matrix tgammamin1 = random(32002,20,1);
> matrix tgamma0 = tkos4pluskos4*tgammamin1;
```

Putting things together, we obtain the desired homomorphism α. It is induced by a homomorphism $\alpha_0 : S(-1)^{10} \to S(-1)^{20} \oplus S$ dual to the map given by the following matrix:

```
> matrix talpha0 = concat(tbeta0,tgamma0);
```

We construct the cokernel of α via a mapping cone:

$$\begin{tikzcd}
M \arrow[d, "\alpha"'] & S(-1)^{10} \arrow[l] \arrow[d, "\alpha_0"'] \arrow[dr, dash, "\oplus" description] & S(-2)^2 \arrow[l, "\varphi"'] \arrow[d, "\alpha_1"'] \arrow[dr, dash, "\oplus" description] & 0 \arrow[l] & \\
N \oplus S & S(-1)^{20} \oplus S \arrow[l] & S(-2)^{20} \arrow[l, "\binom{\psi}{0}"] & S(-3)^{10} \arrow[l] & \cdots \arrow[l]
\end{tikzcd}$$

```
> matrix zero[20][1];
> matrix tpsi = transpose(psi);
> matrix tpresg = concat(tpsi,zero);
> matrix pres = module(transpose(talpha0))
.                      + module(transpose(tpresg));
```

The matrix `pres` is a presentation matrix for $\operatorname{coker}\alpha$, as desired. Minimally resolving its transposed, we get an 11×1 matrix with 1 quartic and 10 quintic entries which define an ideal `I` in `S`:

```
> module dir = transpose(pres);
> dir = prune(dir);
> homog(dir);
1
> intvec deg_dir = attrib(dir,"isHomog");
> attrib(dir,"isHomog",deg_dir-2);         // set degrees
> resolution fdir = mres(prune(dir),2);
> print(betti(fdir),"betti");
            0     1     2
------------------------
   -2:     20    10     -
   -1:      -     1     -
    0:      -     -     -
    1:      -     -     -
    2:      -     -     1
------------------------
total:     20    11     1
> ideal I = groebner(flatten(fdir[2]));
> resolution FI = mres(I,0);
> print(betti(FI),"betti");
            0     1     2     3     4
------------------------------------
    0:      1     -     -     -     -
    1:      -     -     -     -     -
    2:      -     -     -     -     -
    3:      -     1     -     -     -
    4:      -    10    18    10     2
------------------------------------
total:      1    11    18    10     2
```

As in Example 4.13, one can check that `I` defines a smooth surface. □

Remark 4.15 (Further Reading). For some of the basics of homological algebra, see Hilton and Stammbach (1971) and Eisenbud (1995). For more on smooth surfaces in $\mathbb{P}^4$ and their construction, we refer to Decker, Ein, and Schreyer (1993) and Decker and Schreyer (2000).

Lecture 5

Homological Algebra II

In this lecture, we study homological methods in conjunction with several concepts from commutative algebra which are central to algebraic geometry. We begin by introducing flatness and by showing how do check flatness in `SINGULAR`. Then we discuss the relation between depth and codimension and explain how to compute depth via Ext, respectively via the Koszul complex. Finally, we treat Cohen-Macaulay rings from a computational point of view.

5.1 Flatness

The concept of flatness is a riddle that comes out of algebra,
but which is technically the answer to many prayers.

Mumford (1999), Chapter 3, Section 10.

Geometrically, flatness comes into play when studying families of algebraic sets. Naively, we think of such a family as a collection of objects depending on some parameters. The parameters typically vary in an algebraic set, and it is natural to ask that the objects depending on the parameters "vary continuously with the parameters". One attempt of making this precise is to define a family of, say, affine algebraic sets with base T to be an algebraic subset $X \subset T \times \mathbb{A}^n$. Here, the base T is supposed to be an algebraic set, and we think of the fibers of the projection $X \to T$ as the members of the family. A simple example shows that even if we are primarily interested in algebraic sets, schemes occur naturally in this context.

Example 5.1. Consider the family

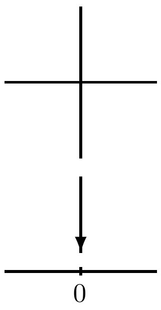
$$\mathrm{V}(x^2 - yt) \subset \mathbb{A}^3 = \mathbb{A}^1_t \times \mathbb{A}^2_{xy} \xrightarrow{\mathrm{pr}} \mathbb{A}^1_t \,.$$

For $t \neq 0$, the fiber over t is a parabola. Over $t = 0$, however, we get the double line $\mathrm{V}(x^2)$. There is nothing wrong with that: as long as we consider the fiber over 0 with its double structure, we may think of it as a conic which is the limit of its neighboring fibers. □

The example suggests that when viewing a morphism of algebraic sets as a family, its members should be the scheme-theoretic fibers rather than the set-theoretic fibers. Without putting further conditions on the morphism, however, the fibers may have little to do with one another. In fact, it may happen that a particular fiber is larger than its neighboring fibers:

Example 5.2. For the first two families below, the dimension of the fiber over 0 exceeds the dimension of its neighboring fibers. For the third one, all fibers are finite, but the fibers over $0, \pm 1$ consist of more points than their neighboring fibers:

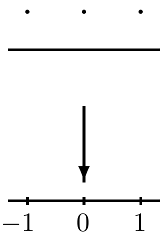
(1)



$$\begin{tikzcd} \mathrm{V}(tx) \subset \mathbb{A}^1_t \times \mathbb{A}^1_x \arrow[d, "\mathrm{pr}"] \\ \mathbb{A}^1_t \end{tikzcd}$$

(2) $\mathrm{V}\big(x^2 - t^{27}yz,\ xz - t^{13}y,\ x - t^{14}z^2\big) \subset \mathbb{A}^1_t \times \mathbb{A}^3_{xyz} \xrightarrow{\mathrm{pr}} \mathbb{A}^1_t.$

(3)



$$\begin{tikzcd} \mathrm{V}(x^2 - x, x(t^3 - t)) \subset \mathbb{A}^1_t \times \mathbb{A}^1_x \arrow[d, "\mathrm{pr}"] \\ \mathbb{A}^1_t \end{tikzcd}$$

Note that for the second family, the fiber over a point $t \neq 0$ is the affine part of a twisted cubic curve, while the fiber over $t = 0$ is a multiple structure on the plane $\mathbb{A}^2 = \mathrm{V}(x) \subset \mathbb{A}^3$. □

In what follows, we will see that the condition of flatness prevents the pathological behavior of the fibers shown in the example above. To begin with, we recall the purely algebraic definition of flatness:

Definition 5.3. A module M over a ring R is called **flat** if for each monomorphism $\iota : N \to L$ of R-modules, the induced map $N \otimes_R M \xrightarrow{\iota \otimes \mathrm{id}_M} L \otimes_R M$ is a monomorphism, too. □

Since tensorizing with a module always preserves right-exact sequences, we may as well say that M is flat over R iff tensorizing with M preserves exact sequences of R-modules. Taking into account that localization preserves exactness and that being injective is a local property of module homomorphisms, we get the following result:

Proposition 5.4 (Flatness is a Local Property). *Let R be a ring, and let M be an R-module. The following are equivalent:*

(1) M is flat over R.
(2) For all prime ideals P of R, the localization M_P is flat over R_P.
(3) For all maximal ideals P of R, the localization M_P is flat over R_P.

Remark 5.5. If R is a graded K-algebra with K a field, and if $\mathfrak{m}$ is the homogeneous maximal ideal of R, then a graded R-module M is flat iff $M_{\mathfrak{m}}$ is flat (see Eisenbud (1995), Exercise 6.10). □

Geometrically, it should be clear from Example 5.1 that the natural environment for introducing flat families is that of schemes. Rather than giving the general definition, however, we focus on some special cases.

Definition 5.6 (Flat Families).

(1) If $\phi : R \to S$ is a homomorphism of rings, we say that ϕ is **flat**, or that S is **flat over R**, if S regarded as an R-module via ϕ is flat.

(2) A morphism $\pi : X \to T$ of affine algebraic sets is **flat** if the induced map of coordinate rings is flat.

(3) If T is an algebraic set, and if $X \subset T \times \mathbb{P}^n$ is an algebraic subset, the projection $\pi : X \to T$ is **flat** if for each point $p \in X$ there are open affine neighborhoods U_p of p in X and $U_{f(p)}$ of $f(p)$ in T such that π restricts to a flat morphism $U_p \to U_{f(p)}$.

(4) We refer to each flat morphism as a **flat family**. □

Example 5.7. The projection in Example 5.2 (1) is not flat. Indeed, the module $M = K[x,t]/\langle tx\rangle$ is not flat over $K[t]$. To see this, consider the inclusion $\iota : \langle t\rangle \to K[x,t]$ of the ideal generated by t in $K[x,t]$ and observe that the induced map

$$\iota \otimes \mathrm{id}_M : \langle t\rangle \otimes_{K[t]} K[x,t]/\langle tx\rangle \to K[x,t] \otimes_{K[t]} K[x,t]/\langle tx\rangle$$

is not injective: $0 \neq t \otimes x \longmapsto t \otimes x = 1 \otimes tx = 0$. □

Remark 5.8 (Flatness and Fibers). For a flat morphism of varieties, all nonempty fibers have the same dimension.[1] See Hartshorne (1977), Corollary

[1] For a flat morphism of *affine* varieties, it may well be that special fibers are empty. For example, consider $\mathrm{V}(xt-1) \subset \mathbb{A}^1_t \times \mathbb{A}^1_x \xrightarrow{\mathrm{pr}} \mathbb{A}^1_t$ (for flatness, argue as in Example 5.14 later in this lecture).

III.9.6 for a more precise statement. Further, flatness implies that several other numerical invariants of the fibers are constant. In fact, if $\pi : X \to T$ is a family as in Definition 5.6 (3) with an irreducible base T, the family is flat iff all scheme-theoretic fibers have the same Hilbert polynomial. See Hartshorne (1977), Theorem III.9.9 and Eisenbud (1995), Exercise 20.14 for more general statements. □

From a computational point of view, we are interested in conditions which characterize flatness, and which can be checked algorithmically. In what follows, we give characterizations in terms of Tor (Theorem 5.11), in terms of syzygies (Theorem 5.12), and in terms of Fitting ideals (Theorem 5.22). While the first two criteria work over *local* Noetherian rings, the third one allows us to check flatness over arbitrary Noetherian rings.

Proposition 5.9 (Flatness via Tor I). *Let R be a ring, and let M be an R-module. Then:*

(1) M is flat iff $\operatorname{Tor}_1^R(R/I, M) = 0$ for each finitely generated ideal $I \subset R$.
(2) If R is a local Noetherian ring with maximal ideal $\mathfrak{m}$, and if M is finitely generated over R, then M is flat over R iff $\operatorname{Tor}_1^R(R/\mathfrak{m}, M) = 0$.

Remark 5.10. Let M be a finitely generated module over an arbitrary ring. Clearly, if M is free, then M is flat. It easily follows from the proposition that over a local Noetherian ring, also the converse is true. □

Statement (1) of Proposition 5.9 can be proved as an application of the long exact sequence of Tor. Note that it is not a practical criterion since it asks us to check a condition for infinitely many ideals. Statement (2) easily follows from Nakayama's lemma. Making use of the Artin-Rees lemma and Krull's intersection theorem, one can show that the assumptions on M can be weakened (see Eisenbud (1995), Theorem 6.8):

Theorem 5.11 (Flatness via Tor II). *Let $(R, \mathfrak{m})$ be a local Noetherian ring, and let $(S, \mathfrak{n})$ be a local Noetherian R-algebra such that $\mathfrak{m}S \subset \mathfrak{n}$. If M is a finitely generated S-module, then M is flat over R iff $\operatorname{Tor}_1^R(R/\mathfrak{m}, M) = 0$.*

In this stronger form, the statement has important applications. For instance:

Theorem 5.12 (Flatness via Syzygies). *Suppose that $(R, \mathfrak{m})$ is a local Noetherian ring, and let $K = R/\mathfrak{m}$ be its residue field. Let $I = \langle f_1, \dots, f_r \rangle$ be an ideal of $R[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle} = R[x_1, \dots, x_n]_{\langle x_1, \dots, x_n \rangle}$, and let $\overline{f}_1, \dots, \overline{f}_r \in K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle}$ be the reductions of $f_1, \dots, f_r$ mod $\mathfrak{m}$. That is, the $\overline{f}_i$ are the images of the f_i under the natural projection $R[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle} \to K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle}$. Then the following are equivalent:*

(1) $R[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle}/I$ is a flat R-module.
(2) $\operatorname{Tor}_1^R(K, R[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle}/I) = 0$.
(3) The syzygies on $\overline{f}_1, \dots, \overline{f}_r$ are generated by the reductions mod $\mathfrak{m}$ of the syzygies on $f_1, \dots, f_r$.

We will refer to condition (3) above by saying that each syzygy on $\overline{f}_1, \ldots, \overline{f}_r$ can be lifted to a syzygy on $f_1, \ldots, f_r$.

Remark 5.13. The statement of Theorem 5.12 also holds if we replace the localization $R[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle}$ by the formal power series ring $R[[\boldsymbol{x}]]$ (and $K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle}$ by $K[[\boldsymbol{x}]]$). Indeed, the proof is word for word identical in both cases, see Greuel and Pfister (2002), Corollary 7.4.7.

Further, if we replace the localization $R[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle}$ by the polynomial ring $R[\boldsymbol{x}]$ (and $K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle}$ by $K[\boldsymbol{x}]$), the implications (1) $\Rightarrow$ (2) $\Leftrightarrow$ (3) remain true. But now, the implication (2) $\Rightarrow$ (1) holds only under some extra assumptions on R. For instance, it holds

(a) if R is a principal ideal domain (for example, if $R = K[t]_{\langle t \rangle}$), or
(b) if R is Artinian, that is, if $\dim R = 0$.

See Eisenbud (1995), Corollary 6.3 for (a), and Artin (1976) for (b).

If $R = K[t_1, t_2]_{\langle t_1, t_2 \rangle}$, however, the implication (2) $\Rightarrow$ (1) does not hold. Indeed, if $I = \langle t_1 x - 1,\, t_2 \rangle \subset R[x]$, the map $\mathfrak{m} \otimes_R R[x]/I \to R \otimes_R R[x]/I$ induced by the inclusion $\mathfrak{m} = \langle t_1, t_2 \rangle \subset R$ is not injective since the nonzero element $t_2 \otimes 1 \in \mathfrak{m} \otimes R[x]/I$ is sent to $t_2 \otimes 1 = 1 \otimes t_2 = 0$. Thus, $R[x]/I$ is not a flat R-module. On the other hand, we have $\operatorname{Tor}_1^R(K, R[x]/I) = 0$. To see this, consider the Koszul complex resolving K:

$$0 \leftarrow K = R/\langle t_1, t_2 \rangle \leftarrow R \xleftarrow{(t_1, t_2)} R^2 \xleftarrow{\binom{-t_2}{t_1}} R \leftarrow 0\,.$$

Then $\operatorname{Tor}_1^R(K, R[x]/I)$ is the homology of the complex

$$R[x]/I \xleftarrow{(t_1, 0)} (R[x]/I)^2 \xleftarrow{\binom{0}{t_1}} R[x]/I$$

which is obviously exact. □

Example 5.14. In Example 5.1, the ideal under consideration is generated by the single polynomial $x^2 - yt$. Reducing mod t, we get the nonzero polynomial x^2 which does not admit a nontrivial syzygy over $K[x, y]$. Thus, the localization of the $K[t]$-module $K[x, y, t]/\langle x^2 - yt \rangle$ at $\langle t \rangle \subset K[t]$ is a flat $K[t]_{\langle t \rangle}$-module by Remark 5.13 (a). The same argument applies to any other maximal ideal $\mathfrak{m} \subset K[t]$ in place of $\langle t \rangle$. Since flatness is a local property (see Proposition 5.4), this implies that $K[x, y, t]/\langle x^2 - yt \rangle$ is a flat $K[t]$-module.

In contrast, we see once more that the $K[t]$-module $K[x, t]/\langle tx \rangle$ in Example 5.2 (1) is not flat. Indeed, the localization of this module at $\langle t \rangle \subset K[t]$ is not a flat $K[t]_{\langle t \rangle}$-module since tx reduces to zero mod t. □

More illustrating is Example 5.2 (2). Reconsidering this example, we will see that the characterization of flatness via syzygies is closely related to the notion of Gröbner bases:

Example 5.15. Let $I \subset \mathbb{Q}[x,y,z,t]$ be the ideal generated by the polynomials

$$f_1 = x^2 - t^{27}yz, \quad f_2 = xz - t^{13}y, \quad f_3 = x - t^{14}z^2.$$

Then $M := \mathbb{Q}[x,y,z,t]/I$ is not a flat $\mathbb{Q}[t]$-module. Indeed, it suffices to show that the localization of M at the maximal ideal $\langle t \rangle \subset \mathbb{Q}[t]$ is not a flat $R := \mathbb{Q}[t]_{\langle t \rangle}$-module. We apply Remark 5.13 (a). Reducing f_1, f_2, f_3 mod $\langle t \rangle$, we get the polynomials

$$\overline{f}_1 = x^2, \quad \overline{f}_2 = xz, \quad \overline{f}_3 = x \in \mathbb{Q}[x,y,z].$$

The syzygies on $\overline{f}_1, \overline{f}_2, \overline{f}_3$ are generated by the column vectors ${}^t(1,0,-x)$ and ${}^t(0,1,-z)$. The first generator can be lifted to a syzygy on f_1, f_2, f_3 :

$$1 \cdot f_1 - x \cdot f_3 = t^{14} z f_2 \,.$$

The situation is different for the second generator:

$$1 \cdot f_2 - z \cdot f_3 = -t^{13}y + t^{14}z^3$$

cannot be written as an $R[x,y,z]$-linear combination of tf_1, tf_2, tf_3. We check this using SINGULAR. As will be explained in Lecture 9, the ring Rxyz below implements $\mathbb{Q}[t]_{\langle t \rangle}[x,y,z]$:

```
> ring Rxyz = 0, (x,y,z,t), (dp(3),ds(1));
```

Note that the chosen monomial order is a mixed order. We compute the syzygies on f_1, f_2, f_3 over $\mathbb{Q}[t]_{\langle t \rangle}[x,y,z]$ and compare their reductions mod $\langle t \rangle$ with the syzygies on $\overline{f}_1, \overline{f}_2, \overline{f}_3$:

```
> ideal I = x2-t27yz, xz-t13y, x-t14z2;
> module M = syz(I);
> print(M);
0,      -1,
z2t14-x,zt14,
xz-yt13,x
> ring Kxyz = 0, (x,y,z), dp;
> ideal I1 = imap(Rxyz,I);
> module M1 = imap(Rxyz,M);  // reducing the syzygies
> M1 = std(M1);
> print(M1);
-z,-1,
x, 0,
0, x
> module M2 = syz(I1);   // syzygies on the reductions
> print(M2);
0, -1,
-1,0,
z, x
> reduce(M2,M1);
_[1]=z*gen(3)-gen(2)
_[2]=0
```

We see that the syzygy ${}^t(0,1,-z)$ cannot be lifted. To analyze this from a Gröbner basis point of view, write $g_i = f_i|_{t=1} \in \mathbb{Q}[x,y,z]$, $i = 1,2,3$, and consider the lexicographic order on $\mathbb{Q}[x,y,z]$. Then $\mathrm{L}(g_i) = \overline{f}_i$ for all i and the two generators for the syzygies on $\overline{f}_1, \overline{f}_2, \overline{f}_3$ correspond to the S-polynomials below:

$$1 \cdot g_1 - x \cdot g_3 = \mathrm{S}(g_1, g_3), \;\; 1 \cdot g_2 - z \cdot g_3 = \mathrm{S}(g_2, g_3) = -y + z^3.$$

That the second syzygy cannot be lifted is reflected by the fact that the condition on remainders in Buchberger's criterion is violated for $\mathrm{S}(g_2, g_3)$. Thus, the set $\{g_1, g_2, g_3\}$ is not a lexicographic Gröbner basis. Adding $g_4 = \mathrm{S}(g_2, g_3)$ to g_1, g_2, g_3, we get such a basis. Correspondingly, if $J \subset \mathbb{Q}[x,y,z,t]$ is the ideal generated by I, and if

$$f_4 := t^{-13} \cdot (1 \cdot f_2 - z \cdot f_3) = -y + tz^3,$$

then the localization of $\mathbb{Q}[x,y,z,t]/J$ at $\langle t \rangle \subset \mathbb{Q}[t]$ is a flat $R = \mathbb{Q}[t]_{\langle t \rangle}$-module:

```
> setring Rxyz;
> ideal J = I, -y+tz3;
> M = syz(J);
> setring Kxyz;
> ideal J1 = imap(Rxyz,J);
> M1 = imap(Rxyz,M); // reducing the syzygies
> M1 = std(M1);
> print(M1);
0, -z,-1,0,
-1,x, 0, 0,
z, 0, x, y,
0, 0, 0, x
> M2 = syz(J1);    // syzygies on the reductions
> print(M2);
0, -1,0,
-1,0, 0,
z, x, y,
0, 0, x
> size(reduce(M2,M1));
0
```

□

The relation between flatness and Gröbner bases is clarified by the following result (which actually holds in a more general context, see Greuel and Pfister (2002), Proposition 7.5.3):

Theorem 5.16 (Flatness and Gröbner Bases). *Let $>$ be a global monomial order on $K[\boldsymbol{x}]$, let $g_1, \dots, g_r \in K[\boldsymbol{x}]$, and let positive weights $w_1, \dots, w_n \in \mathbb{Z}$ be chosen such that, with respect to $\boldsymbol{w} = (w_1, \dots, w_n)$,*

$$\boldsymbol{w}\text{-deg}\big(\mathrm{L}(g_i)\big) > \boldsymbol{w}\text{-deg}\big(\mathrm{L}(\mathrm{tail}(g_i))\big), \quad i = 1, \dots, r\,.$$

Let $>_{\boldsymbol{w}}$ be the weighted degree order on $K[\boldsymbol{x},t]$ with weight vector $(\boldsymbol{w},1)$ extending $>$,[2] *and denote by*

$$g_i^{\text{hom}} := t^{\boldsymbol{w}\text{-deg}(g_i)} \cdot g_i\left(\frac{x_1}{t^{w_1}}, \dots, \frac{x_n}{t^{w_n}}\right)$$

the weighted homogenization *of g_i with respect to t, $i = 1, \dots, r$. Then the following are equivalent:*

(1) $\{g_1, \dots, g_r\}$ is a Gröbner basis with respect to $>$.

(2) $\{g_1^{\text{hom}}|_{t=\lambda}, \dots, g_r^{\text{hom}}|_{t=\lambda}\}$ is a Gröbner basis with respect to $>$ for all $\lambda \in K$.

(3) $\{g_1^{\text{hom}}, \dots, g_r^{\text{hom}}\}$ is a Gröbner basis with respect to $>_{\boldsymbol{w}}$.

(4) $M = K[\boldsymbol{x},t]/\langle g_1^{\text{hom}}, \dots, g_r^{\text{hom}}\rangle$ is a flat $K[t]$-module.

In the situation of the theorem, observe that $g_i^{\text{hom}}|_{t=1} = g_i$ for $i = 1, \dots, r$. Moreover, the weights $w_1, \dots, w_n$ are chosen such that $g_i^{\text{hom}}|_{t=0} = \mathrm{L}(g_i)$. Hence, if I denotes the ideal $I = \langle g_1, \dots, g_r\rangle$ of $K[\boldsymbol{x}]$, and if we view M as a family of affine rings parametrized by the maximal ideals of $K[t]$, the fiber over $\langle t-1\rangle$ is $K[\boldsymbol{x}]/I$, and the fiber over $\langle t\rangle$ is $K[\boldsymbol{x}]/\mathrm{L}(I)$.

Example 5.17. We reconsider Example 5.15 in view of Theorem 5.16.

Let $g_1 = x^2 - yz$, $g_2 = xz - y$, $g_3 = x - z^2$, $g_4 = -y + z^3$, and assign weight 16 to x, weight 4 to y, and weight 1 to z. Then the assumptions of Theorem 5.16 are satisfied for $> = >_{\texttt{lp}}$, and the polynomials $f_1 = x^2 - t^{27}yz$, $f_2 = xz - t^{13}y$, $f_3 = x - t^{14}z^2$, $f_4 = -y + tz^3$ are the weighted homogenizations of $g_1, \dots, g_4$ with respect to t.

As in Example 5.15, let $I = \langle f_1, f_2, f_3\rangle$ and $J = \langle I, f_4\rangle$. Since $\{g_1, g_2, g_3\}$ is not a lexicographic Gröbner basis, the $K[t]$-module $K[x,y,z,t]/I$ is not flat. In contrast, $\{g_1, \dots, g_4\}$ is such a basis, and $K[x,y,z,t]/J$ is flat over $K[t]$.

Geometrically, passing from I to J has no effect on the fibers over a point $t \neq 0$, whereas the fiber over $t = 0$ is cut down from a plane to a line. In fact, if we consider the projective closures, that is, if we homogenize the g_i with respect to a further variable, say w, and if we change the f_i accordingly, then we obtain a flat family whose fiber over a point $t \neq 0$ is a twisted cubic curve, and whose fiber over $t = 0$ consists of a line, a double line, and an embedded point on the double line. Indeed,

$$\langle x^2, xz, wx, w^2y\rangle = \langle x, y\rangle \cap \langle x, w^2\rangle \cap \langle z, x^2, wx, w^2\rangle\,.$$

This multiple structure accounts for the correct Hilbert polynomial:

```
> ring Kwxyz = 0, (w,x,y,z), dp;
> ideal F = x2, xz, wx, w2y;
> displayHilbPoly(std(F));  // enter procedure first
3t+1
```
$\square$

[2] If A is a defining matrix for $>$, then $>_{\boldsymbol{w}}$ is defined by the matrix $\left(\begin{array}{c|c} w_1 \dots w_n & 1 \\ \hline A & 0 \end{array}\right)$.

Remark 5.18. Let $>$ be a global monomial order on $K[\boldsymbol{x}]$, and let $I \subset K[\boldsymbol{x}]$ be an ideal. If I is homogeneous, it follows from Macaulay's Theorem 1.35 that $\dim\big(K[\boldsymbol{x}]/I\big) = \dim\big(K[\boldsymbol{x}]/\mathrm{L}(I)\big)$. As a corollary of Theorem 5.16, one can show that this equality holds for arbitrary ideals I. Further, the theorem and the dimension formula can be formulated such that they hold for arbitrary monomial orders. See Lecture 9, Section 9.4 for more on this. $\square$

Our final characterization of flatness is in terms of Fitting ideals. To begin with, we introduce the necessary notation.

Remark-Definition 5.19. Let R be a Noetherian ring, and let M be a finitely generated R-module. If $0 \leftarrow M \leftarrow R^s \xleftarrow{\varphi} R^t$ is a free presentation of M with presentation matrix φ, we consider the ideals

$$F_i(M) := \big\langle (s-i) \times (s-i) \text{ minors of } \varphi \big\rangle \subset R.$$

Here, we make the convention that $F_i(M) = R$ for $i \geq s$, and that $F_i(M) = 0$ for $i < \max\{s-t, 0\}$. Justifying our notation, we remark that the $F_i(M)$ only depend on M (see Greuel and Pfister (2002), Lemma 7.2.5). We call $F_i(M)$ the ***i*-th Fitting ideal of** M. Note that we have an increasing chain of Fitting ideals

$$0 = F_{-1}(M) \subseteq F_0(M) \subseteq F_1(M) \subseteq \ldots \subseteq F_s(M) = R\,. \qquad \square$$

The SINGULAR procedure `fitting` from `homolog.lib` computes Fitting ideals:

```
proc fitting (matrix phi, int i)
"USAGE:  fitting(phi,i);  phi matrix, i int
RETURN:  ideal, the ith Fitting ideal of M=coker(phi).
"
{
  int s = nrows(phi);
  int a = s-i;
  if (a<=0) { return(ideal(1)); }
  if ( (a>s) or (a>ncols(phi)) ) { return(ideal(0)); }
  return(minor(phi,a));
}
```

To characterize flatness in terms of Fitting ideals, we need the notion of constant rank (see Bruns and Herzog (1993), Section 1.4):

Proposition 5.20. *Let R be a Noetherian ring, let* $\mathrm{Quot}(R)$ *be its total quotient ring, and let M be a finitely generated R-module. The following are equivalent:*

(1) $M \otimes_R \mathrm{Quot}(R)$ is a free $\mathrm{Quot}(R)$-module of rank ρ.
(2) The localization M_P is a free R_P-module of rank ρ for all prime ideals P of R.

Definition 5.21. If M satisfies the equivalent conditions in Proposition 5.20, we say that M has **constant rank** ρ, or simply that M has **rank** ρ. □

If R is a (Noetherian) integral domain, $\mathrm{Quot}(R)$ is a field, and each finitely generated module M over R has constant rank $\dim_{\mathrm{Quot}(R)}\big(M \otimes_R \mathrm{Quot}(R)\big)$. In contrast, if $R = K[x,t]/\langle tx\rangle$, the ideal generated by x in R is not an R-module of constant rank.

Theorem 5.22 (Flatness via Fitting Ideals). *Let R be a Noetherian ring, and let M be a finitely generated R-module. The following are equivalent:*

(1) M is flat and has constant rank ρ.
(2) $F_\rho(M) = R$ and $F_{\rho-1}(M) = 0$.

Of course, the interesting part of this theorem is the implication (2) ⇒ (1). For its proof, notice that if (2) holds, then we may choose an $s \times t$ presentation matrix of M of type

$$\varphi = \begin{pmatrix} E_{s-\rho} & 0 \\ 0 & 0 \end{pmatrix},$$

where E_r is the $r \times r$ identity matrix. Hence, (1) follows as flatness is a local property and as for finitely generated modules over local Noetherian rings, flatness is equivalent to freeness (see Remark 5.10). We refer to Greuel and Pfister (2002), Theorem 7.2.7 for details.

The `SINGULAR` procedure `isFlat` from `homolog.lib` is based on Theorem 5.22:

```
proc isFlat (matrix phi)
"USAGE:   isFlat(phi);  phi matrix
RETURN:  1 if M=coker(phi) is flat;
         0 otherwise.
"
{
   if ( size(ideal(phi))==0 ) { return(1); }
   int i;
   ideal F = fitting(phi,0);
   while ( size(F)==0 )
   {
      i++;
      F = fitting(phi,i);
   }
   if ( deg(std(F)[1])==0 ) { return(1); }
   return(0);
}
```

Example 5.23. (1) We study the projection

$$\pi : \mathrm{V}(x^2 - x, x(t^3 - t)) \subset \mathbb{A}^1_t \times \mathbb{A}^1_x \to \mathbb{A}^1_t$$

from Example 5.2 (3). The $\mathbb{Q}[t]$-module $M := \mathbb{Q}[x,t]/\langle x^2 - x, x(t^3 - t)\rangle$ is generated by (the residue classes of) 1 and x. It has the free presentation

$$0 \longleftarrow M \xleftarrow{(\,1,\ x\,)} \mathbb{Q}[t]^2 \xleftarrow{\binom{0}{t^3-t}} \mathbb{Q}[t]\,.$$

We use `isFlat` to check that M is not flat:

```
> LIB "homolog.lib";
> ring R = 0, t, dp;
> module phi = gen(2)*(t3-t);
> isFlat(phi);
0
```

The procedure `flatLocus` from `homolog.lib` allows us to determine the **flat locus** of M, that is, the set of all prime ideals P of R such that M_P is a flat R_P-module. In fact, if `phi` is a presentation matrix of M, then `flatLocus(phi)` returns the ideal

$$FL(M) = \Big\langle \bigcup_{k\geq 0} \big\{ f \in F_k(M) \mid f \cdot F_{k-1}(M) = 0 \big\} \Big\rangle \subset R\,.$$

It, thus, returns the complement of the flat locus in the sense that

$$M_P \text{ is a flat } R_P\text{-module} \iff P \not\supset FL(M)\,.$$

In the present example, we compute:

```
> flatLocus(phi);
_[1]=t3-t
```

As expected, the result reflects the fact that just the three fibers of the projection π over $t = 0, \pm 1$ are too big. Over $\mathbb{A}^1 \setminus \{0, \pm 1\}$, the family is flat.

(2) Let $I := \langle x^2 - t_1, xt_1 - t_2, xt_2 - t_1^2\rangle \subset \mathbb{Q}[x, t_1, t_2]$, and consider the projection $\pi : \mathrm{V}(I) \subset \mathbb{A}^2_{t_1t_2} \times \mathbb{A}^1_x \to \mathbb{A}^2_{t_1t_2}$.

As above, the $\mathbb{Q}[\boldsymbol{t}] = \mathbb{Q}[t_1, t_2]$-module $M := \mathbb{Q}[x, t_1, t_2]/I$ is generated by 1 and x. To determine a set of generators for the module of syzygies on the generators (and, thus, a free presentation of M), we compute a Gröbner basis for I with respect to the product order $(>_{\mathtt{dp}}, >_{\mathtt{dp}})$ on $\mathbb{Q}[x, \boldsymbol{t}]$:

```
> ring S = 0, (x,t(1..2)), (dp(1),dp);
> ideal I = x2-t(1), x*t(1)-t(2), x*t(2)-t(1)^2;
> std(I);
_[1]=t(1)^3-t(2)^2
_[2]=x*t(2)-t(1)^2
_[3]=x*t(1)-t(2)
_[4]=x^2-t(1)
```

Note that, in terms of the elements of a given Gröbner basis $\mathcal{G} = \{f_1, \dots, f_r\}$ for I with respect to the chosen monomial order, each $\mathbb{Q}[\boldsymbol{t}]$-linear combination of 1 and x in I has a standard expression $\sum_{i=1}^r (a_i \cdot 1 + b_i \cdot x) \cdot f_i$, where $a_i, b_i \in \mathbb{Q}[\boldsymbol{t}]$ such that $b_i = 0$ if $\deg_x(f_i) \geq 1$, and $a_i = 0$ if $\deg_x(f_i) \geq 2$. Hence, from the Gröbner basis computed above, we get the free presentation

$$0 \longleftarrow M \xleftarrow{(1,\, x)} \mathbb{Q}[t_1, t_2]^2 \xleftarrow{\begin{pmatrix} t_1^3-t_2^2 & 0 & -t_1^2 & -t_2 \\ 0 & t_1^3-t_2^2 & t_2 & t_1 \end{pmatrix}} \mathbb{Q}[t_1, t_2]^4 \,.$$

Applying `isFlat`, we see that M and, hence, π is not flat:

```
> ring R1 = 0, t(1..2), dp;
> module phi = gen(1)*(t(1)^3-t(2)^2),
.             gen(2)*(t(1)^3-t(2)^2),
.             gen(2)*t(2)-gen(1)*t(1)^2,
.             gen(2)*t(1)-gen(1)*t(2);
> isFlat(phi);
0
> flatLocus(phi);
_[1]=t(1)^3-t(2)^2
```

This is no surprise as $\mathrm{V}(t_1^3 - t_2^2) \subset \mathbb{A}^2$ is the image of π. Considering π as a map $\mathrm{V}(I) \to \mathrm{V}(t_1^3 - t_2^2)$, we get:

```
> qring Q = std(t(1)^3-t(2)^2);
> module phi = imap(R1,phi);
> isFlat(phi);
0
> flatLocus(phi);
_[1]=t(2)
_[2]=t(1)
```

The result reflects the fact that the fiber of π over $t_1 = t_2 = 0$ is the double point $\mathrm{V}(x^2, t_1, t_2)$, while each of the other fibers over $\mathrm{V}(t_1^3 - t_2^2)$ consists of precisely one reduced point. $\square$

5.2 Depth and Codimension

Throughout this section, we assume that R is a Noetherian ring. We explain how to compute the depth of an ideal I of R. The depth of I is a measure of the size of I, as is its codimension. Though both notions are closely related, the notion of depth is more algebraic and less intuitive than that of codimension. We recall the basic definitions.

Definition 5.24. Let M be an R-module. Then a sequence of nonzero elements $f_1, \dots, f_r \in R$ is called a **regular sequence** on M, or an $\boldsymbol{M}$**-sequence**, if $M \neq \langle f_1, \dots, f_r \rangle M$, and if f_i is a nonzerodivisor on $M / \langle f_1, \dots, f_{i-1} \rangle M$ for $i = 1, \dots, r$. $\square$

Remark 5.25. The permutation of an M-sequence needs not be an M-sequence. See Remark 5.30 and Example 5.32 below. □

Let M be a finitely generated R-module, and let $I \subset R$ be an ideal such that $M \neq IM$. Since R is Noetherian, every M-sequence with elements in I can be extended to a **maximal M-sequence** $f_1, \dots, f_r$ in I. That is, $f_1, \dots, f_r$ is an M-sequence with elements in I such that $f_1, \dots, f_r, g$ is not an M-sequence for all $g \in I$. By exchanging elements in M-sequences, one can show that each maximal M-sequence in I has the same number of elements.[3]

Definition 5.26. Let $I \subset R$ be an ideal, and let M be a finitely generated R-module. If $M \neq IM$, we call

$$\operatorname{depth}(I, M) := \max\big\{r \;\big|\; \text{there is an } M\text{-sequence } f_1, \dots, f_r \in I\big\}$$

the ***I*-depth** of M. We call $\operatorname{depth} I := \operatorname{depth}(I, R)$ the **depth of *I***. If R is a local ring with maximal ideal $\mathfrak{m}$, then $\operatorname{depth}(M) := \operatorname{depth}(\mathfrak{m}, M)$ is called the **depth of *M***. □

Remark 5.27. Let M be a finitely generated R-module, and let $P \subset R$ be a prime ideal containing the **annihilator** of M,

$$\operatorname{ann} M = \{f \in R \mid fM = 0\} = 0 : M \subset R\,.$$

Then each M-sequence in P localizes to an M_P-sequence. Thus,

$$\operatorname{depth}(I, M) \leq \operatorname{depth}(I_P, M_P)$$

for every ideal I of R contained in P. In particular,

$$\operatorname{depth} P \leq \operatorname{depth} R_P$$

for each prime ideal P of R. If P is a maximal ideal of R, equality holds. See Eisenbud (1995), Lemma 18.1 for a proof. □

The following result is the key to computing depth (see Eisenbud (1995), Proposition 18.4 and Greuel and Pfister (2002), Corollary 7.6.11):

Theorem 5.28. *If $I = \langle f_1, \dots, f_r \rangle \subset R$ is an ideal, and if M is a finitely generated R-module such that $M \neq IM$, then*

$$\begin{aligned}\operatorname{depth}(I, M) &= \inf\{k \mid \operatorname{Ext}^k_R(R/I, M) \neq 0\} \\ &= r - \sup\{p \mid H_p(f_1, \dots, f_r; M) \neq 0\}.\end{aligned}$$

Here, we denote by $H_p(f_1, \dots, f_r; M)$ the homology of the Koszul complex at $K(f_1, \dots, f_r; M)_p$.

[3] Other ways of showing this consist of expressing the number of elements of a maximal M-sequence in I in terms of Ext, respectively in terms of the Koszul complex, see Theorem 5.28.

The **Koszul complex** $K(f_1,\dots,f_r;M)$ is defined as the tensor product

$$K(f_1,\dots,f_r;M) = K(f_1,\dots,f_r;R)\otimes_R M\,,$$

where

$$K_p(f_1,\dots,f_r;R) = \bigoplus_{1\le i_1<\dots<i_p\le r} Re_{i_1\dots i_p}$$

is the free R-module of rank $\binom{r}{p}$ with basis vectors $e_{i_1\dots i_p}$, and where the differential $d_p : K_p(f_1,\dots,f_r;R)\to K_{p-1}(f_1,\dots,f_r;R)$ is defined by setting

$$d_p(e_{i_1\dots i_p}) = \sum_{\nu=1}^{p}(-1)^{\nu-1} f_{i_\nu} e_{i_1\dots\widehat{i_\nu}\dots i_p}\,.$$

In particular, this means that $K_0(f_1,\dots,f_r;R)=R$, that d_1 is defined by $d_1(e_i)=f_i$, and that $K_p(f_1,\dots,f_r;R)=0$ if p is not in the range $0\le p\le r$.

Example 5.29. The Koszul complex $K(f_1,f_2,f_3;M)$ reads

$$\begin{tikzcd} M & M^3 \arrow[l, "{(f_1,f_2,f_3)}"'] & M^3 \arrow[l, "{\begin{pmatrix} -f_2 & -f_3 & 0 \\ f_1 & 0 & -f_3 \\ 0 & f_1 & f_2 \end{pmatrix}}"'] & M \arrow[l, "{\begin{pmatrix} f_3 \\ -f_2 \\ f_1 \end{pmatrix}}"'] & 0\,. \arrow[l] \end{tikzcd}$$ □

Remark 5.30. Let $f_1,\dots,f_r$ and M be as in Theorem 5.28. It is easy to check that any permutation of the f_i leads to an isomorphic Koszul complex. Moreover, by construction,

$$H_0(f_1,\dots,f_r;M) = M/\langle f_1,\dots,f_r\rangle M.$$

One can show that if $f_1,\dots,f_r$ is an M-sequence, then all other homology modules of the Koszul complex vanish (that is, the Koszul complex defines a free resolution of $M/\langle f_1,\dots,f_r\rangle M$). In the local, respectively graded case, also the converse is true. See Theorem 5.31 below for a stronger statement (obtained as an application of Nakayama's lemma). In particular, in the situation of Theorem 5.31, any permutation of an M-regular sequence is again an M-regular sequence. □

Theorem 5.31. *Suppose that R is local with maximal ideal $\mathfrak{m}$ (or, that R is a finitely generated graded K-algebra with homogeneous maximal ideal $\mathfrak{m}$) and that $f_1,\dots,f_r$ are (homogeneous) elements of $\mathfrak{m}$. If $0\neq M$ is a finitely generated (graded) R-module, the following are equivalent:*

(1) $f_1,\dots,f_r$ is an M-sequence.
(2) $H_1(f_1,\dots,f_r;M)=0$.

See Matsumura (1986), Theorem 16.5 for a proof.

With respect to computing depth, Theorem 5.28 gives us two ways of how to proceed. One way is based on the computation of Ext, the other relies on the computation of Koszul homology. How to compute Ext was explained in some detail in Lecture 4. Now, we turn to the **computation of Koszul homology**. Let M be a finitely generated R-module, given by a free presentation $0 \leftarrow M \leftarrow R^s \xleftarrow{\varphi} R^t$. Chasing the commutative diagram with exact columns

$$\begin{tikzcd}
 & 0 & 0 & 0 & \\
\cdots & K_{p-1}(\boldsymbol{f};M) \arrow[l] \arrow[u] & K_p(\boldsymbol{f};M) \arrow[l] \arrow[u] & K_{p+1}(\boldsymbol{f};M) \arrow[l] \arrow[u] & \cdots \arrow[l] \\
\cdots & K_{p-1}(\boldsymbol{f};R^s) \arrow[l] \arrow[u] & K_p(\boldsymbol{f};R^s) \arrow[l, "d_p\otimes\mathrm{id}"'] \arrow[u] & K_{p+1}(\boldsymbol{f};R^s) \arrow[l, "d_{p+1}\otimes\mathrm{id}"'] \arrow[u] & \cdots \arrow[l] \\
\cdots & K_{p-1}(\boldsymbol{f};R^t) \arrow[l] \arrow[u, "\mathrm{id}_{p-1}\otimes\varphi"'] & K_p(\boldsymbol{f};R^t) \arrow[l] \arrow[u, "\mathrm{id}_p\otimes\varphi"'] & K_{p+1}(\boldsymbol{f};R^t) \arrow[l] \arrow[u] & \cdots, \arrow[l]
\end{tikzcd}$$

we get

$$H_p(f_1,\dots,f_r;M) = \frac{(d_p\otimes\mathrm{id})^{-1}\big(\mathrm{im}(\mathrm{id}_{p-1}\otimes\varphi)\big)}{\mathrm{im}(d_{p+1}\otimes\mathrm{id}) + \mathrm{im}(\mathrm{id}_p\otimes\varphi)}\,.$$

The `SINGULAR` procedure `KoszulHomology` from `homolog.lib` is based on this formula. Let `f` be an `ideal`, given by an ordered list of polynomials $f_1,\dots,f_r \in R = K[\boldsymbol{x}]$, let `I` be a `module` (corresponding to a submodule I of a free R-module F), and let p be an integer. Then `KoszulHomology(f,I,p)` returns a `module`, say `N`, which is zero if $H_p(f_1,\dots,f_r;F/I) = 0$. If `N` is nonzero, then `matrix(N)` is a presentation matrix of $H_p(f_1,\dots,f_r;F/I)$.

Example 5.32. We use `SINGULAR` to compute the homology of the Koszul complex $K(f_1,f_2,f_3;\mathbb{Q}[x,y,z])$, where $f_1 = (x-1)z$, $f_2 = (x-1)y$ and $f_3 = x$.

```
> LIB "homolog.lib";
> ring R = 0, (x,y,z), dp;
> ideal f = xz-z, xy-y, x;
> module I = 0;             // a presentation matrix of R=Q[x,y,z]
> print(KoszulHomology(f,I,0));             // 0th Koszul homology
z,y,x
> print(KoszulHomology(f,I,1));             // 1st Koszul homology
0
```

From the output, we read $H_0(f_1,f_2,f_3;\mathbb{Q}[x,y,z]) = \mathbb{Q}[x,y,z]/\langle x,y,z\rangle = \mathbb{Q}$, and $H_1(f_1,f_2,f_3;\mathbb{Q}[x,y,z]) = 0$. In contrast to the situation in Theorem 5.31, however, the sequence f_1,f_2,f_3 is not a $\mathbb{Q}[x,y,z]$-sequence since f_2 is a zerodivisor on $\mathbb{Q}[x,y,z]/\langle f_1\rangle$. Indeed, we have $z(xy-y) = y(xz-z)$. On the other hand, the permuted sequence f_1,f_3,f_2 is a $\mathbb{Q}[x,y,z]$-sequence (see also Exercise 3.1). Thus, the first Koszul homology has to vanish, and the Koszul complex $K(f_1,f_2,f_3;\mathbb{Q}[x,y,z])$ is a free resolution of $\mathbb{Q}$. In the following example, obviously no permutation of the given sequence is a $\mathbb{Q}[x,y,z]$-sequence, and the first Koszul homology does not vanish:

```
> ideal g = xy, xz, yz;
> print(KoszulHomology(g,I,0));            // 0th Koszul homology
yz,xz,xy
> print(KoszulHomology(g,I,1));            // 1st Koszul homology
-z,0,x,
z, y,0
> print(KoszulHomology(g,I,2));            // 2nd Koszul homology
0
```
□

Before giving an example of how to compute depth in SINGULAR, we summarize some important facts on depth (see Eisenbud (1995), Corollary 17.8 and Theorem 19.9):

Remark 5.33. (1) Depth is a **geometric notion** in the sense that

$$\operatorname{depth}(I, M) = \operatorname{depth}(\sqrt{I}, M)\,.$$

That is, in case $R = K[\boldsymbol{x}]$, the depth only depends on the algebraic set defined by I.

(2) **Auslander-Buchsbaum Formula.** If R is a local Noetherian ring with maximal ideal $\mathfrak{m}$ (or, if R is a finitely generated graded K-algebra with homogeneous maximal ideal $\mathfrak{m}$), and if M is a finitely generated (graded) R-module of finite projective dimension, then

$$\operatorname{pd}_R(M) = \operatorname{depth}(\mathfrak{m}, R) - \operatorname{depth}(\mathfrak{m}, M)\,.$$

Here, the **projective dimension** of M, written $\operatorname{pd}_R(M)$, is the length of a minimal free resolution of M.

(3) As we already pointed out, the notion of depth is closely related to the notion of codimension. Indeed, if R is any Noetherian ring, and if $I \subsetneq R$ is any ideal, then

$$\operatorname{depth} I \leq \operatorname{codim} I\,.$$

The example below of the homogeneous coordinate ring of the Veronese surface in $\mathbb{P}^4$ shows that the inequality may well be strict. □

The command `depth` from `homolog.lib` relies on the computation of Koszul homology. If `phi` is a `matrix`, corresponding to a presentation matrix φ of a module M, and if `I` is an `ideal`, then `depth(phi,I)` returns $\operatorname{depth}\big(I, M\big)$. Entering `depth(phi);` is equivalent to entering `depth(phi,maxideal(1));`.

Example 5.34. To show `depth` at work, we continue our SINGULAR session from Lecture 4, Example 4.13 in which we created the ring $S = \mathbb{F}_{32003}\,[x_0, \ldots, x_4]$ and an ideal $I \subset S$ defining a Veronese surface in $\mathbb{P}^4$. Now, we compute the depth of the homogeneous maximal ideal $\mathfrak{m}$ of the quotient ring S/I, or, equivalently, the $\langle x_0, \ldots, x_4\rangle$-depth of the S-module S/I:

```
> LIB "homolog.lib";
> depth(I); // I is a presentation matrix of S/I
1
> dim(std(I));
3
```

Hence, $\operatorname{depth}\mathfrak{m} = 1 < 3 = \dim I = \operatorname{codim}\mathfrak{m}$. □

The rings satisfying $\operatorname{depth} I = \operatorname{codim} I$ for each properly contained ideal I are of particular interest. We will study them in the next section.

5.3 Cohen-Macaulay Rings

The notion of Cohen-Macaulay rings is a
workhorse of commutative algebra.
It is sufficiently general to allow a wealth of examples.
It is sufficiently strict to admit a rich theory.
Freely adapted from Bruns and Herzog (1993).

Throughout this section, we assume that R is a Noetherian ring.

Definition 5.35. We say that R is a **Cohen-Macaulay ring** if

$$\operatorname{depth} I = \operatorname{codim} I$$

for every ideal $I \subsetneq R$. □

As one can show, it is enough to ask that $\operatorname{depth} I = \operatorname{codim} I$ for each maximal ideal $I \subset R$.

Example 5.36. If R is zero-dimensional, it is Cohen-Macaulay. The same holds if R is one-dimensional and reduced. If R is one-dimensional but nonreduced, then R may be Cohen-Macaulay or not. The ring $K[x,y]/\langle y^2\rangle$, for instance, is Cohen-Macaulay, while $K[x,y]/\langle xy, y^2\rangle$ is not Cohen-Macaulay. □

The Cohen-Macaulay property is local in a strong sense (see Eisenbud (1995), Proposition 18.8):

Proposition 5.37. *The following are equivalent:*

(1) R is Cohen-Macaulay.
(2) For each maximal ideal $\mathfrak{m} \subset R$, the localization $R_\mathfrak{m}$ is Cohen-Macaulay.
(3) For each maximal ideal $\mathfrak{m} \subset R$, the $\mathfrak{m}$-adic completion $\widehat{R}_\mathfrak{m}$ is Cohen-Macaulay.

If these conditions are satisfied, the localization of R at any prime ideal $P \subset R$ is Cohen-Macaulay, and $\operatorname{depth} P = \operatorname{depth} R_P$.

Remark 5.38. If R is a finitely generated graded algebra over a field K, and if $\mathfrak{m}$ is the homogeneous maximal ideal of R, then R is Cohen-Macaulay iff $R_{\mathfrak{m}}$ is Cohen-Macaulay. □

Remark 5.39. Let $(R, \mathfrak{m})$ be a local ring, and let $f_1, \dots, f_r \in \mathfrak{m}$ be an R-sequence. Then $R/\langle f_1, \dots, f_r \rangle$ is a Cohen-Macaulay ring iff R is a Cohen-Macaulay ring (see Matsumura (1986), Theorem 17.3 (ii)). □

It follows from Krull's principal ideal theorem that if $I \subset R$ is an ideal generated by $c = \operatorname{codim} I$ elements, then I has pure codimension c (that is, all minimal associated primes of I have codimension c).

Definition 5.40. We say that the **unmixedness theorem** holds for R if every ideal $I \subset R$ generated by $\operatorname{codim} I$ elements is unmixed (that is, I has no embedded components). This includes as the case of codimension zero that $\langle 0 \rangle \subset R$ is unmixed. □

The following theorem relates the unmixedness theorem and the Cohen-Macaulay property. For a proof, we refer to Matsumura (1986), Theorem 17.6:

Theorem 5.41. *The unmixedness theorem holds for R iff R is Cohen-Macaulay.*

Macaulay (1916) showed that the unmixedness theorem holds for polynomial rings over a field, and Cohen (1946) proved that it holds for regular local rings. This is the reason for the name Cohen-Macaulay. Taking Theorem 5.41 into account, Macaulay's result is easy to prove. In fact, one shows:

Theorem 5.42. *If R is Cohen-Macaulay, then so is $R[\boldsymbol{x}]$.*

Taking once more Theorem 5.41 into account, Cohen's theorem reads as follows:

Theorem 5.43 (Cohen). *A Noetherian regular local ring is Cohen-Macaulay.*

In this form, the theorem easily follows from the fact that regular local rings are integral domains.

Example 5.44. (1) We say that R is **locally a complete intersection** if for each maximal ideal $\mathfrak{m} \subset R$ there is a Noetherian regular local ring S and an S-sequence $f_1, \dots, f_r \in S$ such that the localization $R_{\mathfrak{m}} \cong S/\langle f_1, \dots, f_r \rangle$. By Cohen's theorem and Remark 5.39, if R is locally a complete intersection, then it is Cohen-Macaulay.

(2) If R is of type $K[\boldsymbol{x}]/I_k(M)$ as in Lecture 2, Theorem 2.25, we refer to it as an affine **determinantal ring**. Theorem 2.25 tells us that such rings are Cohen-Macaulay. □

Remark 5.45. SINGULAR allows us to check the Cohen-Macaulay property for rings of type $K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle}/I$, where $I \subset K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle}$ is an ideal given by finitely many polynomial generators. Having implemented $K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle}$ using a local monomial order as explained in Lecture 9, and having created the `ideal I`, we may check the defining condition of a Cohen-Macaulay ring by computing depth either directly or via the Auslander-Buchsbaum formula:

(1) Entering

```
depth(I) == dim(std(I));
```

SINGULAR will display `1`, if equality holds (that is, if $K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle}/I$ is Cohen-Macaulay), and `0`, otherwise. The command `isCM` from `homolog.lib` is based on this approach.

(2) Using `mres`, we may easily write a procedure `projdim` which takes as input the ideal `I`, and which returns the projective dimension of $K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle}/I$. Then we may enter

```
nvars(basering)-projdim(I) == dim(std(I));
```

Taking Remark 5.38 into account, the Cohen-Macaulay property for a graded affine ring $K[\boldsymbol{x}]/I$ may be checked by applying the above to the ideal generated by I in $K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle}$. □

Example 5.46. We check that the homogeneous coordinate ring of the twisted cubic curve is Cohen-Macaulay. The local ring $K[x,y,z,w]_{\langle x,y,z,w \rangle}$ is implemented using a local monomial order (see Lecture 9 for details).

```
> LIB "homolog.lib";
> ring R = 0, (x,y,z,w), dp;
> ideal I = xz-y2, wz-xy, wy-x2;
> ring R_loc = 0, (x,y,z,w), ds;
> ideal I = imap(R,I);
> isCM(I);
1
```
□

Observe that is often much easier to check whether the following sufficient conditions are satisfied.

Proposition 5.47. *Let $I \subset K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle}$ be an ideal given by finitely many polynomial generators, let $>$ be a local monomial order on $K[\boldsymbol{x}]$, and let $\mathrm{L}(I) \subset K[\boldsymbol{x}]$ be the leading ideal of I as defined in Lecture 9. Suppose that the following conditions hold for some c:*

(1) $\mathrm{L}(I) \supset \langle x_1, \dots, x_c \rangle^N$ *for some N;*
(2) $\mathrm{L}(I)$ *is generated by monomials in $K[x_1, \dots, x_c]$.*

Then $K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle}/I$ is a Cohen-Macaulay ring.

Remark 5.48. The statement of Proposition 5.47 also holds if we replace $K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle}/I$ and the local monomial order by a graded affine ring $K[\boldsymbol{x}]/I$ and a global monomial order. □

Example 5.49. Now using Remark 5.48, we check once more that the homogeneous coordinate ring of the twisted cubic curve is Cohen-Macaulay:

```
> ring R = 0, (x,y,z,w), dp;
> ideal I = xz-y2, wz-xy, wy-x2;
> I = std(I);
> lead(I);             // the leading ideal
_[1]=y2
_[2]=xy
_[3]=x2
```
□

Note that the assumptions of Proposition 5.47 imply that $S = K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle}/I$ is a finitely generated free (hence, flat) $R = K[x_{c+1}, \dots, x_n]_{\langle x_{c+1},\dots,x_n \rangle}$-module. Thus, the proposition follows from the first part of the following theorem which associates the Cohen-Macaulay property with flatness.

Theorem 5.50. *Let* $(R, \mathfrak{m})$ *and* $(S, \mathfrak{n})$ *be Noetherian local rings, and let* $\phi : R \to S$ *be a* **homomorphism of local rings** *(that is,* $\phi(\mathfrak{m}) \subset \mathfrak{n}$*). Then:*

(1) If S is flat over R, the following are equivalent:
  (a) S is Cohen-Macaulay;
  (b) R and $S/\phi(\mathfrak{m})S$ are Cohen-Macaulay.

(2) If R is regular and if S is Cohen-Macaulay, the following are equivalent:
  (a) $\dim(S) = \dim(R) + \dim(S/\phi(\mathfrak{m})S)$*;*
  (b) S is flat over R. □

For a proof, we refer to Matsumura (1986), Theorem 23.1 and Corollary 23.3, respectively to Eisenbud (1995), Theorem 18.16.

In the following geometric interpretation of Theorem 5.50, we say that an algebraic set A is **locally Cohen-Macaulay** if all its local rings $\mathcal{O}_{A,p}$ are Cohen-Macaulay.

Remark 5.51. Let $\pi : X \to T$ be a morphism as in Definition 5.6 (2) or (3). Then, as flatness is a local property, Theorem 5.50 implies:

(1) If π is flat, then X is locally Cohen-Macaulay iff the base T and all fibers are locally Cohen-Macaulay. In particular, if π is flat and X is locally Cohen-Macaulay, the scheme-theoretic fibers of π have no embedded components.
(2) If X is locally Cohen-Macaulay and connected, and if T is smooth and connected, then π is flat iff all isolated components of each fiber are of dimension $\dim(X) - \dim(T)$ (see Remark 5.8).

In Example 5.1, $X = \mathrm{V}(x^2 - yt)$ is locally Cohen-Macaulay of dimension 2, and each fiber is irreducible of dimension $1 = \dim(X) - \dim(\mathbb{A}^1_t)$. So (2) shows once more that the projection $X \to \mathbb{A}^1_t$ is flat. □

Remark 5.52 (Further Reading). For more details and complete proofs of the results presented in this lecture, see Matsumura (1986), Eisenbud (1995), Chapters 17–20, and Greuel and Pfister (2002), Chapter 7. For more on the richness of the theory of Cohen-Macaulay rings, we refer to Bruns and Herzog (1993).

Practical Session III

Exercise 3.1. Write a `SINGULAR` procedure which checks whether a given (ordered) list of polynomials $f_1, \ldots, f_r \in K[x_1, \ldots, x_n]$ defines a regular sequence on $K[x_1, \ldots, x_n]$. Apply this procedure to the polynomials $f_1 = (x-1)z$, $f_2 = (x-1)y,\ f_3 = x \in \mathbb{Q}[x, y, z]$ and permutations thereof.

Exercise 3.2. Use `SINGULAR` to check whether the following rings are Cohen–Macaulay:

(a) $\mathbb{C}[x, y, z]/\langle xy, yz, xz\rangle$;
(b) the homogeneous coordinate ring $\mathbb{C}[s^4, s^3t, st^3, t^4]$ of the rational quartic curve in $\mathbb{P}^3$.

Exercise 3.3. Let $M = \bigoplus_{\nu \in \mathbb{Z}} M_\nu$ be a finitely generated graded module over $K[x_1, \ldots, x_n]$.

(a) For $d \in \mathbb{Z}$, the **truncated module** $M_{\geq d}$ is defined to be the graded submodule
$$M_{\geq d} = \bigoplus_{\nu \geq d} M_\nu \subset M.$$
Write a `SINGULAR` procedure which, given a presentation matrix of M, computes a presentation matrix of $M_{\geq d}$.

(b) The Castelnuovo-Mumford **regularity** of M is defined to be the least integer r such that, for every i, the ith syzygy module of M is generated in degrees $\leq r+i$ (in particular, M is generated in degrees $\leq r$). In terms of the graded Betti numbers of M, we ask that $\max\{j \mid \beta_{ij}(M) \neq 0\} \leq r+i$ for every i. Write a `SINGULAR` procedure which, given a presentation matrix of M, computes the regularity of M.

(c) Let $R = \mathbb{Q}[w, x, y, z]$, let I be the graded submodule of $F = R \oplus R(-1)^2 \oplus R(-2)^2$ considered in Section 3.4 of Lecture 3, and let $M = F/I$. Compute the regularity of M and the minimal free resolutions of the truncated modules $M_{\geq d}$ for $d = 2, 3$.

Exercise 3.4. Write `SINGULAR` procedures for computing kernels of module homomorphisms, Hom and Ext over $K[x_1, \ldots, x_n]$.

Exercise 3.5. Consider the Koszul complex over $S = K[x_0, \ldots, x_4]$ resolving K (choose $K = \mathbb{F}_{32003}$ for your computations). Use `SINGULAR` to check that the cokernel of a "generic" homomorphism

$$\mathcal{S}(-1)^5 \longrightarrow \mathrm{Syz}_3(K(2))$$

is the ideal of a smooth surface in $\mathbb{P}^4$. Do the same for a "generic" homomorphism

$$M = \mathrm{Syz}_4(K(3))^3 \longrightarrow N,$$

where N is the kernel of a "generic" homomorphism

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

Compare the Betti diagram obtained by resolving the ideal of the first surface with that of the surface `QES` constructed in Exercise 1.5.

Lecture 6

Solving Systems of Polynomial Equations

Depending on the application one has in mind, "**solving**" a system

$$\begin{aligned} f_1(x_1,\dots,x_n) &= 0 \\ &\vdots \\ f_r(x_1,\dots,x_n) &= 0 \end{aligned} \tag{6.1}$$

of polynomial equations could mean to decide whether there are finitely many solutions and, if so, to find all solutions (or, to find all solutions together with their multiplicities). It could mean to find just one solution, or to represent each irreducible component of the solution set by explicit points on it. We may be interested in finding the solutions over the given coefficient field K, or over some extension field of K. And, if K is a subfield of $\mathbb{C}$, we may aim at representing the solutions symbolically, or at computing floating point approximations of the solutions up to a given precision.

In this lecture, we pay special attention to the latter problem, which is widely considered to be a task just for numerical analysis. However, numerical methods are often unstable in an unpredictable way, they may have problems with over-determined systems (that is, systems with more equations than variables), and they may not find all solutions, respectively the correct number of different real or complex solutions[1].

Taking our cue from this situation, we explore the possibility of using Gröbner bases in a **symbolic-numerical approach** to solving. Such an approach combines symbolic and numerical methods with the aim of determining the solutions more accurately. Here, the symbolic methods are used to find additional information on the structure of the solution set which allows one some control of the numerical methods and/or to preprocess the given system of equations such that it is expected to be better suited for numerical methods (see Fig. 6.1 for an example).

[1] See Cox, Little, and O'Shea (1998), Chapter 2 for some examples of what can happen.

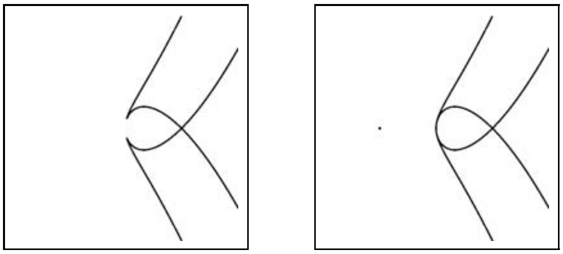


Fig. 6.1. Purely numerical methods for visualizing the set of real solutions of a polynomial equation $f(x_1, x_2) = 0$ may run into trouble near singular points, and they may fail to find isolated solutions (left). A symbolic-numerical approach can produce relief here (right)

Another topic in this lecture is a classical symbolic approach to solving which makes use of resultants and which can be combined with numerical methods, too (we will not discuss numerical methods, here).

Even though our main interest lies in a symbolic-numerical approach, we set up most of the theory behind the symbolic methods for an arbitrary coefficient field K. In principle, this provides the basis for finding symbolic solutions for the system (6.1) over the algebraic closure of K.

6.1 Gröbner Basis Techniques

Let K be an arbitrary field. We consider an ideal $I = \langle f_1, \dots, f_r\rangle \subset K[\boldsymbol{x}]$ and its vanishing locus $\mathrm{V}(I) \subset \mathbb{A}^n = \mathbb{A}^n(\overline{K})$, where $\overline{K}$ is an algebraically closed extension field of K. If $p \in \mathrm{V}(I)$ is a point, we occasionally refer to it as a **solution of *I***.

6.1.1 Computing Dimension

Let $>$ be a global monomial order on $K[\boldsymbol{x}]$, and let $\mathcal{G}$ be a Gröbner basis for I with respect to $>$. Since

$$\dim(K[\boldsymbol{x}]/I) = \dim(K[\boldsymbol{x}]/\mathrm{L}_>(I)) \tag{6.2}$$

by Lecture 5, Remark 5.18, we may compute the dimension of $\mathrm{V}(I)$ starting from $\mathcal{G}$ by purely combinatorial means.

To explain this in detail, we begin by recalling that, due to Hilbert's Nullstellensatz, $\mathrm{V}(I)$ is nonempty iff no element of $\mathcal{G}$ has 1 as its leading monomial. Moreover, if $\mathrm{V}(I)$ is nonempty and finite, say

$$\mathrm{V}(I) = \{(a_{j,1}, \dots, a_{j,n}) \mid 1 \le j \le s\} \subset \mathbb{A}^n \,,$$

the products

$$g_i := \prod_{j=1}^{s} (x_i - a_{j,i}) \in \overline{K}[x_i]\,, \quad i = 1, \dots, n\,,$$

vanish along $\mathrm{V}(I)$. Applying again the Nullstellensatz, we see that $g_i^{m_i} \in I\,\overline{K}[\boldsymbol{x}]$ for some $m_i \in \mathbb{N}$, so $x_i^{sm_i} \in \mathrm{L}_>(I)$ (it is clear from what we said in Lecture 2, Remark 2.9 on the role of the coefficient field that each generating set of monomials for $\mathrm{L}_>(I)$ is also a generating set for $\mathrm{L}_>(I\,\overline{K}[\boldsymbol{x}])$. Combining this with Macaulay's Theorem 1.33 in Lecture 1, we get:

Theorem 6.1 (Conditions for Finiteness). *The following are equivalent:*

(1) $\mathrm{V}(I)$ *is a finite subset of* $\mathbb{A}^n$.
(2) For each $1 \le i \le n$, *there is some* $\alpha_i \in \mathbb{N}$ *such that* $x_i^{\alpha_i} \in \mathrm{L}_>(I)$.
(3) For each $1 \le i \le n$, *there is some* $\alpha_i \in \mathbb{N}$ *such that* $x_i^{\alpha_i}$ *is the leading monomial of some element of the Gröbner basis* $\mathcal{G}$.
(4) The K*-vector space* $K[\boldsymbol{x}]/\mathrm{L}_>(I)$ *is finite dimensional.*
(5) The K*-vector space* $K[\boldsymbol{x}]/I$ *is finite dimensional.*

Moreover, if $I \subsetneq K[\boldsymbol{x}]$ *is proper, then (1) – (5) are equivalent to* $\dim I = 0$.

Remark 6.2 (Multiplicities). Suppose that $\mathrm{V}(I)$ is finite. Given a point $p \in \mathrm{V}(I)$, there is a natural way of assigning a multiplicity to p as a solution of I, written $\mathrm{mult}\,(p \mid I)$ (see Definition 9.32 in Lecture 9). Counted with multiplicity, there are precisely $\dim_K(K[\boldsymbol{x}]/I) = \dim_K(K[\boldsymbol{x}]/\mathrm{L}_>(I))$ solutions of I over $\overline{K}$ (see Lecture 9, Remark 9.33). That is, counted with multiplicity, the number of solutions of I equals the number of standard monomials for I. Note that all multiplicities are 1 iff $I\overline{K}[\boldsymbol{x}]$ is a radical ideal. □

Turning to the computation of dimension in general, we note that if $I \subsetneq K[\boldsymbol{x}]$ is any proper ideal, then

$$\dim K[\boldsymbol{x}]/I = \dim \overline{K}[\boldsymbol{x}]/I\,\overline{K}[\boldsymbol{x}] = \dim \mathrm{V}(I)\,. \tag{6.3}$$

For the second equality, recall from Lecture 2 that by its very definition, $\dim \mathrm{V}(I)$ is the dimension of the vanishing ideal of $\mathrm{V}(I)$ which in turn is the dimension of the radical of $I\overline{K}[\boldsymbol{x}]$ by Hilbert's Nullstellensatz. Then note that the dimension of an ideal coincides with that of its radical. For the first equality, see Remark 2.9 in Lecture 2. Alternatively, observe that the dimension of a monomial ideal $J \subset K[\boldsymbol{x}]$ is particularly easy to determine. Indeed, if $g_1, \dots, g_r$ are monomial generators for J, then $\dim J$ is the maximal cardinality of a subset $\boldsymbol{u} \subset \boldsymbol{x}$ such that none of the g_i is in $K[\boldsymbol{u}]$. Together with what we said right before Theorem 6.1, this gives

$$\dim K[\boldsymbol{x}]/\mathrm{L}_>(I) = \dim \overline{K}[\boldsymbol{x}]/\mathrm{L}_>(I\,\overline{K}[\boldsymbol{x}])\,,$$

and we are done by equation (6.2).

Summing up our discussion, we get the following recipe for computing dimension in the general case:

Theorem 6.3. *If $I \subsetneq K[\boldsymbol{x}]$ is a proper ideal, then* $\dim I = \dim \mathrm{V}(I)$ *is the maximal cardinality of a subset $\boldsymbol{u} \subset \boldsymbol{x}$ such that $\mathrm{L}_>(I) \cap K[\boldsymbol{u}] = \{0\}$. That is, if $\mathcal{G}$ is a Gröbner basis for I (with respect to any global monomial order on $K[\boldsymbol{x}]$), then* $\dim I = \dim \mathrm{V}(I)$ *is the maximal cardinality of a subset $\boldsymbol{u} \subset \boldsymbol{x}$ such that none of the leading terms of the elements of $\mathcal{G}$ is in $K[\boldsymbol{u}]$.*

See Greuel and Pfister (2002), Theorem 3.5.1 and Exercise 3.5.1 and Decker and Schreyer (2006), Chapter 3 for alternative proofs of this theorem.

6.1.2 Zero-Dimensional Solving by Elimination

Throughout this section, we suppose that I is a zero-dimensional ideal, that is, $\mathrm{V}(I)$ is a (nonempty) finite subset of $\mathbb{A}^n$. A *naive idea* for finding the points of $\mathrm{V}(I)$ originates from Theorem 6.1:

Remark 6.4. In the situation of Theorem 6.1, if we choose $>$ to be a global elimination order with respect to $\boldsymbol{x} \setminus \{x_i\}$, condition (2) of the theorem implies that the principal ideal $I \cap K[x_i]$ is nontrivial. Computing a Gröbner basis for I with respect to such an order, we get a generator, say h_i, for $I \cap K[x_i]$. Rather than performing n different Gröbner basis computations, we may obtain the h_i simultaneously starting from a single Gröbner basis for I (with respect to an arbitrary monomial order) by making use of the linear algebra behind the FGLM algorithm (see Lecture 3, Section 3.5 for that algorithm). The `SINGULAR` implementation of this idea is accessible by the `finduni` command which takes as input a reduced Gröbner basis for I. □

In principle, we may, thus, compute the points of $\mathrm{V}(I)$ as follows:

Step 1. Compute generators h_i for $I \cap K[x_i]$, $i = 1, \ldots, n$.

Step 2. Solve the univariate equations $h_i(x_i) = 0$ over $\overline{K}$ to obtain candidates for each single coordinate entry of a point of $\mathrm{V}(I)$.

Step 3. Form all points with coordinates as computed in Step 2, substitute these points into the equations to distinguish solutions from nonsolutions, and discard nonsolutions.

Here, in Step 2, either use a numerical solver or compute the roots of the h_i symbolically by factorizing the h_i (see Example 6.16).

For another approach to solving, recall that the lexicographic order $>_{\mathtt{lp}}$ has the elimination property with respect to any initial set of variables. In conjunction with Theorem 6.1, this gives:

Theorem 6.5. *Let $\mathcal{G}$ be a Gröbner basis for the zero-dimensional ideal I with respect to $>_{lp}$. Then there are elements $g_1, \ldots, g_n \in \mathcal{G}$ such that*

$$\begin{array}{ll} g_1 \in K[x_n]\,, & \mathrm{L}(g_1) = c_1 x_n^{m_1}, \\ g_2 \in K[x_{n-1}, x_n]\,, & \mathrm{L}(g_2) = c_2 x_{n-1}^{m_2}, \\ \vdots & \quad\vdots \\ g_n \in K[x_1, \ldots, x_n]\,, & \mathrm{L}(g_n) = c_n x_1^{m_n}\,. \end{array} \tag{6.4}$$

Thus, starting from a lexicographic Gröbner basis $\mathcal{G}$ for I, computing $\mathrm{V}(I)$ is again reduced to univariate solving: we solve $g_1(x_n) = 0$, substitute the solutions for x_n in $g_2(x_{n-1}, x_n) = 0$, solve the resulting univariate equation, and so on. In a last step, we discard those points of $\mathrm{V}(g_1, \dots, g_n)$ at which at least one of the polynomials in $\mathcal{G} \setminus \{g_1, \dots, g_n\}$ does not vanish.

Having to detect nonsolutions in a last step makes the two methods for solving just discussed especially prone to errors introduced by numerical root finding. This applies, specifically, to the second method. Indeed, substituting approximate solutions into an equation makes the equation itself only approximate and rounding errors will accumulate. In Section 6.1.3 below, refining the second method, we discuss algorithms for triangular decomposition which allow us to reduce the zero-dimensional solving problem to the problem of solving several zero-dimensional systems with n equations in n variables of the form (6.4) in Theorem 6.5. In this way, the step in which nonsolutions have to be detected becomes superfluous.

Remark 6.6 (Zero-Dimensional Radical). If K is a perfect field, and if $h_1, \dots, h_n$ are polynomials as in Remark 6.4, then the radical $\sqrt{I}$ is generated by I and the square-free parts of the h_i (see, for instance, Greuel and Pfister (2002), Chapter 4). In practical applications, the square-free parts are obtained by gcd computations (see, for instance, Geddes, Czapor, and Labahn (1992)). With respect to solving, if one is not interested in computing the multiplicities of the solutions, replacing I by $\sqrt{I}$ is one possible way of preprocessing (see Example 6.14 later in this lecture). This is of particular importance for a symbolic-numerical approach since multiple solutions may cause severe problems for numerical solvers. $\square$

6.1.3 Decomposition

Given an ideal $I = \langle f_1, \dots, f_r \rangle \subset K[\boldsymbol{x}]$ of arbitrary dimension and its vanishing locus $\mathrm{V}(I) \subset \mathbb{A}^n = \mathbb{A}^n(\overline{K})$, decomposition techniques aim at computing a list of finite sets of polynomials $\mathcal{F}_1, \dots, \mathcal{F}_\ell \subset K[\boldsymbol{x}]$ such that

$$\mathrm{V}(I) = \mathrm{V}(\mathcal{F}_1) \cup \ldots \cup \mathrm{V}(\mathcal{F}_\ell)\,.$$

One way of doing this is to compute the minimal associated primes of I (more generally, to compute a primary decomposition of I). We will discuss symbolic algorithms for primary decomposition in Lecture 7. Note, however, that these algorithms are usually not suited for symbolic preprocessing as they are too expensive.[2]

In this section, our focus is on cheaper decomposition techniques. We mainly discuss algorithms for triangular decomposition which we combine

[2] See Sommese and Wampler (2005) for a purely numerical approach to primary decomposition and solving.

with numerical root finding to obtain a symbolic-numerical method for solving zero-dimensional systems. We begin, however, by explaining the factorizing Buchberger algorithm which in some examples finds a nontrivial decomposition rather quickly, while in other examples it spends its time trying to factorize irreducible polynomials.

The Factorizing Buchberger Algorithm

As its name suggests, the factorizing Buchberger algorithm combines Buchberger's algorithm with (multivariate) polynomial factorization. Applied to the generators $f_1, \ldots, f_r$ for I, it computes a list of Gröbner bases $\mathcal{G}_1, \ldots, \mathcal{G}_\ell$ such that

$$\mathrm{V}(I) = \mathrm{V}(\mathcal{G}_1) \cup \ldots \cup \mathrm{V}(\mathcal{G}_\ell)\,. \tag{6.5}$$

In general, the $\mathrm{V}(\mathcal{G}_i)$ are not irreducible.

The basic idea of the algorithm is easy to describe. Apply Buchberger's algorithm to $f_1, \ldots, f_r$. Whenever a nonzero remainder h appears in Buchberger's test, apply a polynomial factorization algorithm to h. If this yields a nontrivial factorization, say $h = h_1 h_2$ decomposes into a product of two factors, apply the factorizing Buchberger algorithm recursively to both ideals $\langle I, h_1\rangle$ and $\langle I, h_2\rangle$ (where I is represented by the Gröbner basis elements computed so far). Geometrically, we have a decomposition

$$\mathrm{V}(I) = \mathrm{V}(I, h_1) \cup \mathrm{V}(I, h_2).$$

This decomposition may be redundant in that $\mathrm{V}(I, h_1)$ and $\mathrm{V}(I, h_2)$ may have some irreducible components in common. One way to overcome this problem is to replace $\langle I, h_2\rangle$ by the saturation $\langle I, h_2\rangle : \langle h_1\rangle^\infty$. Indeed,

$$\mathrm{V}(I) = \mathrm{V}(I, h_1) \cup \mathrm{V}\big(\langle I, h_2\rangle : \langle h_1\rangle^\infty\big) = \mathrm{V}(I, h_1) \cup \overline{\mathrm{V}(I, h_2) \setminus \mathrm{V}(h_1)}$$

is an irredundant decomposition. However, applying saturation at each step may be costly. An alternative approach, which reduces some of the redundancy, is to use h_1 as a **nonzero constraint** for the factorizing Buchberger algorithm applied to $\langle I, h_2\rangle$. That is, compute a standard expression for h_1 in terms of the given generators for $\langle I, h_2\rangle$. If the resulting remainder is zero, there is no need to consider $\langle I, h_2\rangle$ any further. Otherwise, proceed similarly for each new factor found in subsequent steps. See Czapor (1989) and Gräbe (1995a) for more details.

Remark 6.7 (The `facstd` Command). `SINGULAR` offers an implementation of the factorizing Buchberger algorithm which is accessible via the `facstd` command. This command takes as input two ideals, say `I` and `J`. The second input parameter `J`, which is optional, implements additional nonzero constraints (the given generators for `J`). If `facstd(I,J)` returns the list $\mathcal{G}_1, \ldots, \mathcal{G}_\ell$ of Gröbner bases, then

$$\mathrm{V}(I) \setminus \mathrm{V}(J) \subset \mathrm{V}(\mathcal{G}_1) \cup \ldots \cup \mathrm{V}(\mathcal{G}_\ell) \subset \mathrm{V}(I)\,.$$

If $\mathcal{G}_1, \ldots, \mathcal{G}_\ell$ happen to generate prime ideals of $K[\boldsymbol{x}]$, we have

$$\mathrm{V}(\mathcal{G}_1) \cup \ldots \cup \mathrm{V}(\mathcal{G}_\ell) = \overline{\mathrm{V}(I) \setminus \mathrm{V}(J)}\,.$$

In general, however, this is not the case (see Example 6.8 below).

Note that the number of sets in the decomposition computed by `facstd` may depend on some random choices made by the polynomial factorization algorithm (these choices affect the order in which the factors are returned). □

Example 6.8. We give an example of a zero-dimensional ideal I in a polynomial ring with 5 variables to indicate that usually the number of sets in the decomposition computed by `facstd` depends on the choice of monomial order. We begin by picking the lexicographic order:

```
> option(redSB);
> ring R = 0, x(1..5), lp;
> poly f1 = x(1)^2+x(1)+2*x(2)*x(5)+2*x(3)*x(4);
> poly f2 = 2*x(1)*x(2)+x(2)+2*x(3)*x(5)+x(4)^2;
> poly f3 = 2*x(1)*x(3)+x(2)^2+x(3)+2*x(4)*x(5);
> poly f4 = 2*x(1)*x(4)+2*x(2)*x(3)+x(4)+x(5)^2;
> poly f5 = 2*x(1)*x(5)+2*x(2)*x(4)+x(3)^2+x(5);
> ideal I = f1^2,f2^2,f3,f4,f5;
> list L = facstd(I);
> size(L);     // number of sets in the decomposition
12
```

Inspecting the entries of `L`, we see that they represent maximal ideals of $\mathbb{Q}[\boldsymbol{x}]$. Thus, if $\overline{\mathbb{Q}}$ is the algebraic closure of $\mathbb{Q}$, each entry of `L` defines a set of points in $\mathbb{A}^n(\overline{\mathbb{Q}})$ which are pairwise conjugate over $\mathbb{Q}$.[3] In fact, each entry `L[i]` consists of a set of 5 polynomials of the form (6.4) in Theorem 6.5. For instance, `L[3]` defines a single, rational point:

```
> L[3];
_[1]=5*x(5)-1
_[2]=5*x(4)-1
_[3]=5*x(3)-1
_[4]=5*x(2)-1
_[5]=5*x(1)+4
```

Applying `facstd(I,5*x(5)-1)` instead of `facstd(I)`, we get:

```
> list L2 = facstd(I,5*x(5)-1);
> size(L2);
11
```

[3] In general, if $\overline{K}$ is the algebraic closure of K, and if P is a maximal ideal of $K[\boldsymbol{x}]$, the vanishing locus $\mathrm{V}(P) \subset \mathbb{A}^n(\overline{K})$ is an orbit under the natural action of the Galois group of $\overline{K}$ over K on $\mathbb{A}^n(\overline{K})$. We then say that the points of $\mathrm{V}(P)$ are pairwise **conjugate over $\boldsymbol{K}$**.

Inspecting the entries of `L2`, we see that they describe $\mathrm{V}(I) \setminus \mathrm{V}(5x_5 - 1)$.

Whereas with respect to $>_{\mathtt{lp}}$, `facstd` separates all components over $\mathbb{Q}$, it shows the opposite extremal behavior when applied with respect to $>_{\mathtt{dp}}$:

```
> ring R1 = 0, x(1..5), dp;
> ideal I = imap(R,I);
> size(facstd(I));
1
> size(facstd(I,5*x(5)-1));
1
```

Actually, one can check that in this case the results returned by `facstd(I)` and `facstd(I,5*x(5)-1)` coincide. □

Triangular Decompositions

Let $I \subset K[\boldsymbol{x}] = K[x_1, \dots, x_n]$ be a zero-dimensional ideal.

Definition 6.9. A set of polynomials $\mathcal{T} = \{g_1, \dots, g_n\} \subset K[\boldsymbol{x}]$ is called a **triangular basis** if, for each $j = 1, \dots, n$,

(TB 1) $g_j \in K[x_{n-j+1}, \dots, x_n] \setminus K$, and

(TB 2) the leading monomial of g_j with respect to the lexicographic order is of the form $x_{n-j+1}^{m_j}$ for some $m_j \geq 1$.

A list of triangular bases $\mathcal{T}_1, \dots, \mathcal{T}_\ell$ is called a **triangular decomposition** for I if $\mathrm{V}(I) = \mathrm{V}(\mathcal{T}_1) \cup \ldots \cup \mathrm{V}(\mathcal{T}_\ell)$. □

Note that the product criterion implies that a triangular basis $\mathcal{T}$ is a (minimal) lexicographic Gröbner basis (of a zero-dimensional ideal). Hence, "triangular basis" can be understood as a short-cut for "triangular lexicographic Gröbner basis". The key point is that the number of elements of such a basis coincides with the number of variables. Thus, if we solve the corresponding zero-dimensional system using the second approach to zero-dimensional solving described in Section 6.1.2, the step in which we have to detect nonsolutions becomes superfluous. As an immediate consequence, we obtain:

Remark 6.10 (Number of Solutions). If $\mathcal{T} = \{g_1, \dots, g_n\} \subset K[\boldsymbol{x}]$ is a triangular basis, then Remark 6.2 implies that, counted with multiplicity, the number of solutions of $g_1 = \cdots = g_n = 0$ over the algebraic closure of K is the product of the degrees $\deg(\mathrm{L}_{>_{\mathtt{lp}}}(g_i))$, $i = 1, \dots, n$. □

The `SINGULAR` library `triang.lib` provides implementations of two different algorithms which use Gröbner bases to compute triangular decompositions of zero-dimensional ideals. These algorithms are due to Lazard (1992), respectively Möller (1993).

We study in some detail a variant of the algorithm by Möller as implemented in `SINGULAR` by Hillebrand (1999). This algorithm returns an irredundant triangular decomposition. Starting from a reduced lexicographic Gröbner basis $\mathcal{G} = \{f_1, \ldots, f_s\}$ for I, sorted such that $\mathrm{L}_{>_{\mathrm{lp}}}(f_s) > \ldots > \mathrm{L}_{>_{\mathrm{lp}}}(f_1)$, it recursively computes a decomposition of the form

$$\mathrm{V}(\mathcal{G}) = \dot{\bigcup_{i=1}^{s-1}} \Big(\underbrace{\mathrm{V}(\mathcal{G}, a_1, \ldots, a_{i-1}) \setminus \mathrm{V}(a_i)}_{=:\, W_i}\Big) \,\dot\cup\, \Big(\mathrm{V}(\mathcal{G}, a_1, \ldots, a_{s-1})\Big), \qquad (6.6)$$

where $a_i \in K[x_2, \ldots, x_n]$ is the leading coefficient of f_i as a univariate polynomial in x_1. Note that the sets W_i are Zariski closed (since $\mathrm{V}(\mathcal{G})$ is finite). A set of defining equations for W_i can be computed from $\mathcal{G} \cup \{a_1, \ldots, a_{i-1}\}$ by saturating with respect to $\langle a_i \rangle$.

Two key observations make this approach work: from our assumptions, it follows that

(1) $\mathrm{L}_{>_{\mathrm{lp}}}(f_s) = x_1^m$, for some $m \geq 1$, and that

(2) $\{a_1, \ldots, a_{s-1}\}$ is a lexicographic Gröbner basis for the ideal quotient $J := \langle f_1, \ldots, f_{s-1}\rangle : \langle f_s \rangle \subset K[\boldsymbol{x}]$. In particular,

$$\mathrm{V}(\mathcal{G}, a_1, \ldots, a_{s-1}) = \mathrm{V}(\underbrace{a_1, \ldots, a_{s-1}}_{\subset K[x_2, \ldots, x_n]}, f_s)\,. \qquad (6.7)$$

Since $\mathcal{G}$ is a reduced lexicographic Gröbner basis, (1) is an immediate consequence of Theorem 6.5. Moreover, for each $1 \leq i \leq s-1$, we have $\mathrm{L}_{>_{\mathrm{lp}}}(f_i) = \mathrm{L}_{>_{\mathrm{lp}}}(a_i) \cdot x_1^{d_i}$, for some $0 \leq d_i < m$.

To see (2), fix the lexicographic order on $K[\boldsymbol{x}]$ and write $\mathrm{L} = \mathrm{L}_{>_{\mathrm{lp}}}$. Buchberger's criterion implies that each S-polynomial $\mathrm{S}(f_i, f_j)$, $1 \leq j < i \leq s-1$, has a standard expression in terms of $f_1, \ldots, f_s$ with remainder 0. Since these S-polynomials have degree at most $m-1$ as a univariate polynomial in x_1, the summand involving f_s in such a standard expression is zero. Thus, once more applying Buchberger's criterion, we get that $\{f_1, \ldots, f_{s-1}\}$ is a lexicographic Gröbner basis, too. It follows that, for each $g \in J$, the product $g \cdot f_s \in \langle f_1, \ldots, f_{s-1}\rangle$ has a standard expression in terms of $f_1, \ldots, f_{s-1}$ with remainder 0. In particular, $\mathrm{L}(gf_s) = \mathrm{L}(g) \cdot x_1^m$ is divisible by some $\mathrm{L}(f_i) = \mathrm{L}(a_i) \cdot x_1^{d_i}$, $1 \leq i \leq s-1$. We conclude that $\mathrm{L}(g)$ is divisible by $\mathrm{L}(a_i)$, and it only remains to show that $\{a_1, \ldots, a_{s-1}\} \subset J$. Fix some $i \leq s-1$. The difference $a_i f_s - x_1^{m-d_i} f_i \in I$ has degree at most $m-1$ as a univariate polynomial in x_1 and, thus, a standard expression in terms of $f_1, \ldots, f_{s-1}$ with remainder 0 (argue as above). It follows that $a_i f_s \in \langle f_1, \ldots, f_{s-1}\rangle$, that is, $a_i \in J$.

Since the ideal $I' := \langle a_1, \ldots, a_{s-1}\rangle \subset K[x_2, \ldots, x_n]$ is again zero-dimensional, the equality (6.7) allows us to compute a triangular decomposition of the right-most set in the decomposition (6.6) by recursion on the number of variables: If a triangular decomposition $\mathcal{T}'_1, \ldots, \mathcal{T}'_{\ell'}$ for I' is given, then $\mathcal{T}'_1 \cup \{f_s\}, \ldots, \mathcal{T}'_{\ell'} \cup \{f_s\}$ is a triangular decomposition for $\langle I', f_s\rangle \subset K[\boldsymbol{x}]$. Summing up, we get:

Algorithm 6.11 (`triangMH`).

```
INPUT:   a list of polynomials $f_1, \dots, f_s \in K[\boldsymbol{x}] = K[x_1, \dots, x_n]$ forming
         a reduced lexicographic Gröbner basis for a zero-dimensional
         ideal of $K[\boldsymbol{x}]$.
ASSUME:  $\mathrm{L}_{>_{\mathrm{lp}}}(f_s) > \dots > \mathrm{L}_{>_{\mathrm{lp}}}(f_1)$.
OUTPUT:  a list $\mathcal{T}_1, \dots, \mathcal{T}_\ell$ of triangular bases such that $\mathrm{V}(f_1, \dots, f_s)$ is
         the disjoint union of $\mathrm{V}(\mathcal{T}_1), \dots, \mathrm{V}(\mathcal{T}_\ell)$.
```

```
Step 1 (Initialization):
  • set $\mathcal{G}_1 := \{f_1, \dots, f_s\}$ ;
  • for $i = 1, \dots, s-1$, let $a_i \in R := K[x_2, \dots, x_n]$ be the leading coefficient
    of $f_i$ considered as an element of $R[x_1]$ ;
  • reduce the lexicographic Gröbner basis $\{a_1, \dots, a_{s-1}\}$ and assign the re-
    sult to $\mathcal{G}'$;

Step 2 (Recursion in Dimension):
  • compute $L' :=$ triangMH($\mathcal{G}'$)  (in $n-1$ variables) ;
  • define $L$ to be the list of all $\mathcal{T}' \cup \{f_s\}$, $\mathcal{T}' \in L'$ ;

Step 3 (Iteration): for $i = 1, \dots, s-1$ do
    if $a_i \notin \langle \mathcal{G}_i \rangle$
        • compute the reduced lexicographic Gröbner basis $\mathcal{G}_i'$ for the satu-
          ration $\langle \mathcal{G}_i \rangle : \langle a_i \rangle^\infty$;
        • append the result of triangMH($\mathcal{G}_i'$) to $L$;
        • set $\mathcal{G}_{i+1} := \mathcal{G}_i \cup \{a_i\}$;
    else set $\mathcal{G}_{i+1} := \mathcal{G}_i$ ;

return(L).
```
□

Remark 6.12. If $\mathcal{G}$ is the reduced lexicographic Gröbner basis for I, and if $\mathcal{T}_1, \dots, \mathcal{T}_\ell$ are the triangular bases computed by applying `triangMH` to $\mathcal{G}$, then $I \subset \langle \mathcal{T}_1 \rangle \cap \dots \cap \langle \mathcal{T}_\ell \rangle \subset \sqrt{I}$. Note that both inclusions may well be strict. □

Example 6.13. We continue the `SINGULAR` session from Example 6.8, now computing a triangular decomposition for the zero-dimensional ideal I under consideration:

```
> LIB "triang.lib";
> setring R;
> ideal G = groebner(I);  // option(redSB) is already set
> list T = triangMH(G); T;
[1]:
   _[1]=x(5)
   _[2]=x(4)
```

```
     _[3]=x(3)
     _[4]=x(2)^2
     _[5]=x(1)^4+2*x(1)^3+x(1)^2
  [2]:
     _[1]=9765625*x(5)^10-1
     _[2]=x(4)-15625*x(5)^7
     _[3]=x(3)-25*x(5)^3
     _[4]=x(2)^2-781250*x(2)*x(5)^9+15625*x(5)^8
     _[5]=2*x(1)+31250*x(2)*x(5)^6+625*x(5)^5+1
  [3]:
     _[1]=95367431640625*x(5)^20-1201171875*x(5)^10+1
     _[2]=11*x(4)^2-1281738281250*x(4)*x(5)^17[...]
     _[3]=11*x(3)+152587890625*x(4)*x(5)^16-1906250*x(4)*x(5)^6[...]
     _[4]=22*x(2)+275*x(4)*x(5)^2+16021728515625*x(5)^19[...]
     _[5]=22*x(1)+3814697265625*x(4)*x(5)^18-47656250*x(4)*x(5)^8[...]
```

Thus, the computed triangular decomposition of I consists of three triangular bases. □

Together with what we did in Section 6.1.2, we are now in the position to formulate a **symbolic-numerical algorithm for zero-dimensional solving**:

INPUT:	a list of polynomials $f_1, \dots, f_r \in K[\boldsymbol{x}]$ generating a zero-dimensional ideal of $K[\boldsymbol{x}]$, K a subfield of $\mathbb{C}$.
OUTPUT:	the set of all complex solutions of $f_1 = \ldots = f_r = 0$ (the multiplicities of the solutions are neglected).

Step 1. Compute a reduced lexicographic Gröbner basis for $I = \langle f_1, \dots, f_r \rangle$.

Step 2. Starting from the Gröbner basis, compute a triangular decomposition $\mathcal{T}_1, \dots, \mathcal{T}_\ell$ for I using Algorithm 6.11.

Step 3. For each i, successively use a numerical solver to find the coordinate entries of the complex zeros of $\mathcal{T}_i$ as explained in Section 6.1.2.

A `SINGULAR` implementation of this algorithm is accessible via the `solve` command provided by `solve.lib`. To get more reliable numerical solutions, however, it may be preferable to proceed step by step, inspecting the triangular bases obtained, and applying additional symbolic preprocessing steps, if needed.

Example 6.14. We continue our session from Example 6.13 in which we already computed a triangular decomposition `T` for the ideal `I` under consideration. Now, we apply Step 3 above. For this, we make use of the `triang_solve` command which is also provided by the library `solve.lib`:

```
  > LIB "solve.lib";
  > triang_solve(T,20);      // 20 digits should be displayed
  // ** Laguerre: Too many iterations!
  // ** rootContainer::solver: No roots found!
```

We see that the numerical solver has problems to find the roots. Therefore, some further symbolic preprocessing is advised. As a next step, we replace each triangular basis of `T` by a triangular basis representing the radical of the ideal generated by the basis. Here, we make use of the `zeroRad` command, which is based on Remark 6.6, and which is provided by the library `primdec.lib`. Further, we factorize the first, univariate, polynomials in two of the triangular bases.[4] Here, we apply the `factorize` command which decomposes a polynomial over the given coefficient field (see Lecture 7, Remark 7.1 for more on this command). Each factor leads to a new triangular basis. We group these bases together in the new triangular decomposition `TS`:

```
> LIB "primdec.lib";
> list TR;
> int k,j;
> for (k=1; k<=size(T); k++)
.   { TR = TR + triangMH(std(zeroRad(T[k]))); }
> triang_solve(TR,20);
// ** Laguerre: Too many iterations!
// ** rootContainer::solver: No roots found!
> for (k=1; k<=size(TR); k++) { print(TR[k][1]); }
x(5)
9765625*x(5)^10-1
95367431640625*x(5)^20-1201171875*x(5)^10+1
> list TS = TR[1];
> ideal J,JJ,LL;
> for (k=2; k<=size(TR); k++)
. {  J = TR[k];
.    LL = factorize(J[1],2)[1]; // returns nonconstant factors only
.    for (j=1; j<=size(LL); j++)
.    {  JJ = LL[j],J[2..size(J)];
.       TS = TS+list(JJ);  }
. }
> size(TS);      // number of triangular bases
11
> def RC = triang_solve(TS,20);
> setring RC;
> basering;
//   characteristic : 0 (complex:20 digits, additional 20 digits)
//   1 parameter    : I
//   minpoly        : (I^2+1)
//   number of vars : 5
//        block   1 : ordering lp
//                  : names    x(1) x(2) x(3) x(4) x(5)
//        block   2 : ordering C
> size(rlist);   // number of complex solutions
32
```

[4] Since the coefficients of these polynomials differ in size by a factor of 10^7, respectively of 10^{14}, they are not well-suited for direct numerical solving.

```
> rlist[3];      // the third solution
[3]:
   [1]:
      -0.8
   [2]:
      (-0.16180339887498948482-I*0.11755705045849462583)
   [3]:
      (0.06180339887498948482+I*0.19021130325903071442)
   [4]:
      (0.06180339887498948482-I*0.19021130325903071442)
   [5]:
      (-0.16180339887498948482+I*0.11755705045849462583)
```
□

Having computed a triangular decomposition, we can, in principle, also make use of it to find symbolic solutions. Typically, this requires to work over (successive) finite extensions of the given coefficient field.

Remark 6.15 (Finite Field Extensions in SINGULAR). Recall that each finite extension L of a perfect field K is **simple**, that is, L is obtained from K by adjoining a root α of a (monic) irreducible polynomial $h \in K[t]$. As usual, we refer to α as a **primitive element** and to h as the **minimal polynomial** of the extension. Given a prime field K and a tower of finite extensions

$$K \subset K(\alpha_1) \subset K(\alpha_1, \alpha_2) \subset K(\alpha_1, \alpha_2, \dots, \alpha_s) = L\,,$$

at this writing, the extension field L can only be implemented in SINGULAR if its minimal polynomial over K is known. Given the minimal polynomial of each extension $K(\alpha_1, \dots, \alpha_{i+1}) \supset K(\alpha_1, \dots, \alpha_i)$, the minimal polynomial of $L \supset K$ can be computed with SINGULAR by using the command `primitive` from `primitiv.lib`. □

We illustrate symbolic solving by a small example:

Example 6.16. We use SINGULAR to compute symbolic solutions for the following zero-dimensional system of polynomial equations which is given by a triangular basis:

$$z^5 + z^3 - 2z^2 - 2 \;=\; y^2 + z^2 + 1 \;=\; x^2 + 2yx - x - y - z^2 - 3 \;=\; 0\,.$$

```
> ring R  = 0, (x,y,z), lp;
> ideal I = z5+z3-2z2-2, y2+z2+1, x2+2yx-x-y-z2-3;
```

We begin by factorizing the univariate polynomial `I[1]`:

```
> def F = factorize(I[1],1); F;
F[1]=z3-2
F[2]=z2+1
```

Treating the second irreducible factor of `I[1]` first, we now choose $\mathbb{Q}(i) = \mathbb{Q}[t]/\langle t^2 + 1\rangle$ as our coefficient field and substitute i for z in I:

```
> ring R1 = (0,a), (x,y), lp;
> map phi = R,x,y,a;
> minpoly = number(phi(F)[2]);
> ideal Iz = phi(I);              // substitute a for z
> Iz = simplify(Iz,2); Iz;        // remove zero generators
Iz[1]=y2
Iz[2]=x2+2*xy-x-y-2
```

From the output, we read that the solutions with z-component i have y-component 0 (and multiplicity at least 2). Substituting 0 for y, we can compute the x-coordinates of these solutions:

```
> ideal Izy = reduce(Iz,std(y)); // substitute 0 for y
> Izy = simplify(Izy,2); Izy;
Izy[1]=x2-x-2
> def Fzy = factorize(Izy[1],1); Fzy;
Fzy[1]=x-2
Fzy[2]=x+1
```

So far, we got the four solutions $(2, 0, \pm i)$, $(-1, 0, \pm i)$ of I (each of these solutions has multiplicity 2). We continue by considering the first irreducible factor $z^3 - 2$ of `I[1]`. Proceeding as above, we get

```
> ring R2 = (0,b), (x,y), lp;
> map phi = R,x,y,b;
> minpoly = number(phi(F)[1]);
> ideal Iz = phi(I);              // substitute a for z
> Iz = simplify(Iz,2); Iz;        // remove zero generators
Iz[1]=y2+(b2+1)
Iz[2]=x2+2*xy-x-y+(-b2-3)
> def Fz = factorize(Iz[1],1);  Fz;
Fz[1]=y2+(b2+1)
```

The output shows that we have to adjoin $i\sqrt{(\sqrt[3]{2})^2 + 1}$ to the coefficient field $\mathbb{Q}[t]/\langle t^3 - 2\rangle = \mathbb{Q}(\sqrt[3]{2})$. As discussed in Remark 6.15, we apply the command `primitive` from `primitiv.lib`:

```
> LIB "primitiv.lib";
> ring S = 0, (b,c), dp;
> ideal E = b3-2, c2+(b2+1);
> def L = primitive(E); L;
L[1]=c6+3c4+3c2+5
L[2]=1/2c4+c2+1/2
L[3]=c
```

The entry `L[1]` is the minimal polynomial of the simple extension and `L[2]`, `L[3]` represent $\sqrt[3]{2}$, $i\sqrt{(\sqrt[3]{2})^2 + 1}$.

```
> ring R3 = (0,c), x, lp;
> def L = imap(S,L);
> map phi = R, x, L[3], L[2];
> minpoly = number(L[1]);
> ideal Izy = simplify(phi(I),2); Izy;
Izy[1]=x2+(2c-1)*x+(c2-c-2)
> def Fzy = factorize(Izy[1],1);  Fzy;
Fzy[1]=x+(c+1)
Fzy[2]=x+(c-2)
```

Thus, we found four more solutions of I:

$$\left(-1 \mp i\sqrt{(\sqrt[3]{2})^2+1},\ \pm i\sqrt{(\sqrt[3]{2})^2+1},\ \sqrt[3]{2}\right),$$
$$\left(2 \mp i\sqrt{(\sqrt[3]{2})^2+1},\ \pm i\sqrt{(\sqrt[3]{2})^2+1},\ \sqrt[3]{2}\right).$$

We leave it as an exercise to compute the remaining eight solutions (corresponding to the remaining two zeros of z^3-2). This requires a further field extension. □

Remark 6.17. Triangular decompositions can be defined and computed for ideals of positive dimension, too (see Aubry, Lazard, and Moreno Maza (1999), and Gräbe (1995b)). For a comparison of triangular decomposition methods which do not rely on Gröbner basis computations, see Aubry and Moreno Maza (1999). □

6.2 Resultant Based Methods

In this section, we briefly discuss a classical approach to the elimination problem which makes use of resultants. Before turning to the general concept of multipolynomial resultants and their application to the solving problem, we consider the well-known Sylvester resultant of two univariate polynomials.

6.2.1 The Sylvester Resultant

If R is a ring, and if

$$f=\sum_{i=0}^{d} a_i x^i\,,\quad g=\sum_{j=0}^{e} b_j x^j \in R[x]$$

are two polynomials of positive degrees d, e, the **Sylvester resultant** of f and g is the determinant

$$\operatorname{Res}(f,g) := \det \begin{pmatrix} a_d & a_{d-1} & \dots & a_0 & & & \\ & a_d & a_{d-1} & \dots & a_0 & & \\ & & \ddots & & & \ddots & \\ & & & a_d & a_{d-1} & \dots & a_0 \\ b_e & b_{e-1} & \dots & \dots & b_0 & & \\ & \ddots & & & & \ddots & \\ & & b_e & b_{e-1} & \dots & \dots & b_0 \end{pmatrix} \in R\,.$$

Here, the matrix has e rows containing a_i's and d rows containing b_j's.

The Sylvester resultant has the following **key properties**:

(R1) *If R is a unique factorization domain, then* $\operatorname{Res}(f,g) = 0$ *iff f and g have a common nonconstant factor in $R[x]$.*

(R2) *If R is an integral domain, then* $\operatorname{Res}(f,g) = Af + Bg$ *for some polynomials $A, B \in R[x]$.*

Remark 6.18. Note that the Sylvester resultant is an integer polynomial in the coefficients of f and g. That is, there is a polynomial

$$\operatorname{Res}_{(d,e)} \in \mathbb{Z}[u_0, \dots, u_d, v_0, \dots, v_e]$$

such that if f, g are as above, then

$$\operatorname{Res}(f,g) = \operatorname{Res}_{(d,e)}(a_0, \dots, a_d, b_0, \dots, b_e)\,.$$

Similarly, we may think of A and B in (R2) as integer polynomials in the coefficients of f and g. □

The property (R2) implies that $\operatorname{Res}(f,g) \in \langle f, g\rangle \cap R$. As demonstrated by the following example, this makes the resultant applicable to elimination problems:

Example 6.19. Consider the polynomials

$$f = xy^2 - xy - y^3 + 1\,, \quad g = x^2y^2 - x^2y + xy - 1 \in \mathbb{Q}[x,y]\,.$$

Here, we face two variables, but we can "hide" the y-variable by regarding f and g as univariate polynomials in x with coefficients in $R = \mathbb{Q}[y]$. Then

$$\operatorname{Res}(f,g) = \det \begin{pmatrix} y^2 - y & -y^3 + 1 & 0 \\ 0 & y^2 - y & -y^3 + 1 \\ y^2 - y & y & -1 \end{pmatrix}$$
$$= y^8 - y^7 + y^6 - 3y^5 + y^4 + y^3 + y^2 - y\,.$$

We compute this resultant and its irreducible factors using `SINGULAR`:

```
> ring S = 0, (x,y), dp;
> poly f, g = xy2-xy-y3+1, x2y2-x2y+xy-1;
> poly r = resultant(f,g,x); r;
y8-y7+y6-3y5+y4+y3+y2-y;
> factorize(r,2);     // display nonconstant factors only
[1]:
   _[1]=y-1
   _[2]=y
   _[3]=y5+y4+2y3-y-1
[2]:
   2,1,1
```

As stated above, $\mathrm{Res}(f,g)$ is contained in the elimination ideal $\langle f,g\rangle \cap \mathbb{Q}[y]$. It follows that the y-coordinates of all complex solutions of $f(x,y)=g(x,y)=0$ must be zeros of $\mathrm{Res}(f,g) = y(y-1)^2(y^5+y^4+2y^3-y-1)$. That is, if $\pi : \mathrm{V}(f,g) \to \mathbb{A}^1$ is the projection which sends (a,b) to b, then

$$\pi(\mathrm{V}(f,g)) \subset \mathrm{V}(\mathrm{Res}(f,g)) .$$

To determine the points of $\mathrm{V}(\mathrm{Res}(f,g))$ which are not in the image of π (if any), observe that if none of the leading coefficients of $f,g \in R[x]$ vanishes at $y_0 \in \mathbb{A}^1$, then the resultant of the univariate polynomials $f(x,y_0)$, $g(x,y_0) \in \mathbb{C}[x]$ coincides with $\mathrm{Res}(f,g)(y_0)$. In our example, this shows that $\pi(\mathrm{V}(f,g))$ contains $\mathrm{V}\big(\mathrm{Res}(f,g)\big) \setminus \mathrm{V}(y^2-y)$ (use property (R1) of the resultant).

It, thus, remains to check the points in $\mathrm{V}(y^2-y)$. Since $f(x,1)=0$, $g(x,1)=x-1$, we get that $y_0=1$ is in the image of π. In contrast, $f(x,0)=1$ is a nonzero constant, so $y_0=0$ is not the y-coordinate of a complex solution of $f(x,y)=g(x,y)=0$. In fact, $y_0=0$ is not a zero of the generator of the elimination ideal $I_1 = \langle f,g\rangle \cap \mathbb{Q}[y]$:

```
> ideal I1 = eliminate(ideal(f,g),x);
> I1;
I1[1]=y6+y4-2y3-y2+1
> factorize(I1[1],2);
[1]:
   _[1]=y-1
   _[2]=y5+y4+2y3-y-1
[2]:
   1,1
```
□

The considerations made in the example lead to the following remark:

Remark 6.20. Let $f=\sum_{i=0}^{d} a_i x^i$, $g=\sum_{j=0}^{e} b_j x^j$ be polynomials of positive degrees d,e in x, with coefficients $a_i, b_j \in R = K[\boldsymbol{y}] = K[y_1,\dots,y_m]$. Further, suppose that a_d and b_e are (nonzero) constants. Then

$$\mathrm{V}\big(\mathrm{Res}(f,g)\big) = \mathrm{V}\big(\langle f,g\rangle \cap K[\boldsymbol{y}]\big) \subset \mathbb{A}^m .$$

Note, however, that even in this case, $\langle \mathrm{Res}(f,g)\rangle$ may be a proper subideal of the elimination ideal $\langle f,g\rangle \cap K[\boldsymbol{y}]$:

```
> ring S = 0, (x,y), dp;
> poly f, g = x2+y2-1, x2+2y2-1;
> resultant(f,g,x);
y4
> eliminate(ideal(f,g),x);
_[1]=y2
```
□

Actually, using generalized resultants as in Example 6.24 below, one can prove the following theorem (see Cox, Little, and O'Shea (1997), Chapter 3, §6):

Theorem 6.21 (Extension Theorem). *Let $I = \langle f_1, \dots, f_r\rangle \subset K[\boldsymbol{x}]$, let $1 \le k \le n$, and let $\mathcal{G}_{k-1}$ be a lexicographic Gröbner basis for the elimination ideal I_{k-1}. Regard the elements of $\mathcal{G}_{k-1}$ as univariate polynomials in $K[x_{k+1}, \dots, x_n][x_k]$. If the leading coefficients of the elements of $\mathcal{G}_{k-1}$ do not all vanish at*

$$(a_{k+1}, \dots, a_n) \in \mathrm{V}(I_k) \subset \mathbb{A}^{n-k},$$

then there is some $a_k \in \mathbb{A}^1$ such that

$$(a_k, \dots, a_n) \in \mathrm{V}(I_{k-1}).$$

In the situation of the theorem, we also say that $(a_{k+1}, \dots, a_n)$ is a **partial solution of I**.

Remark 6.22. Theorem 6.21 yields an alternative method for zero-dimensional solving by elimination. Indeed, after a general change of coordinates, the assumption of Theorem 6.21 is satisfied for all k (if K is infinite, the change of coordinates can be taken to be linear; see, for instance, Decker and Schreyer (2006), Chapter 3). We may, then, proceed similar to Section 6.1.2, successively extending partial solutions to solutions. Since each partial solution can be extended, we do not have to discard nonsolutions in a final step. Nevertheless, from a practical point of view, this method is usually too expensive. Applying a general coordinate transformation destroys sparseness and makes all subsequent computations much more involved. □

In the next two examples, we use resultants to compute defining equations for parametrized curves. The first example is taken from Cattani and Dickenstein (2005). It shows that resultant based methods may behave much better than Gröbner basis techniques.[5]

Example 6.23 (Fröberg). We compute a defining equation for the affine plane curve given by the polynomial parametrization

$$t \mapsto \left(t^{32},\ t^{48} - t^{56} - t^{60} - t^{62} - t^{63}\right)\ :$$

[5] Via a Hilbert driven approach, we also succeeded to compute the desired equation using the `eliminate` command. On its way, `SINGULAR` allocated more than 7 GB of memory, and it took more than 2 days of computing to get the result.

```
> ring R = 0, (x,y,t), dp;
> poly f = x-t32;
> poly g = y-t48+t56+t60+t62+t63;
> int aa = timer;
> poly h = resultant(f,g,t);
> timer-aa;
2
> h;
x63-595965x62-32x61y+6143174x61+3656768x60y+464x59y2-70859517x60
-65651616x59y-13277840x58y2-4064x57y3+49771514x59+220805184x58y+
[...]
-448x8y27-88x7y28-120x6y28+32x5y29+16x3y30-y32
> deg(h);                    // the total degree
63
> size(h);                   // number of terms of h
257
```

The result of the computation is a polynomial `h` of degree 63 which is the degree expected by Bézout's Theorem. It follows that `h` generates the elimination ideal $\langle f, g\rangle \cap \mathbb{Q}[x,y]$, where $f = x - t^{32}$, $g = y - t^{48} + t^{56} + t^{60} + t^{62} + t^{63}$. $\square$

Example 6.24. We compute the equations of the affine twisted cubic curve $C \subset \mathbb{A}^3 = \mathbb{A}^3(\mathbb{C})$ from its parametrization $t \mapsto \big(t, t^2, t^3\big)$. Now, we face the problem of eliminating t from the ideal of $\mathbb{Q}[x,y,z,t]$ generated by the *three* polynomials $f_1 = x - t$, $f_2 = y - t^2$, $f_3 = z - t^3$, and we cannot apply the Sylvester resultant directly. To overcome this problem, we introduce auxiliary variables u_2, u_3 and compute

$$h = \operatorname{Res}\big(f_1, u_2 f_2 + u_3 f_3\big) \in \mathbb{Q}[x,y,z,u_2,u_3] \; :$$

```
> ring R = 0, (x,y,z,t,u(2),u(3)), dp;
> poly f(1), f(2), f(3) = x-t, y-t2, z-t3;
> poly h = resultant(f(1), u(2)*f(2)+u(3)*f(3), t);
> h;
x^3*u(3)+x^2*u(2)-y*u(2)-z*u(3)
```

Considering $h = h(u_2, u_3)$ as an element of $\mathbb{Q}[x,y,z][u_2,u_3]$, we refer to its coefficients as **generalized resultants**. If $a, b \in \mathbb{Q}$, $b \neq 0$, then $h(a,b)$ coincides with the resultant of f_1 and $af_2 + bf_3$. It is, thus, an element of the elimination ideal $\langle f_1, af_2 + bf_3\rangle \cap \mathbb{Q}[x,y,z]$. In particular, for all $(a,b) \in \mathbb{Q}^2$ with $b \neq 0$, $h(a,b)$ vanishes along $C = \mathrm{V}(\langle f_1, f_2, f_3\rangle \cap \mathbb{Q}[x,y,z])$. This implies that each of the generalized resultants vanishes along C. We compute:

```
> ideal CO = coeffs(coeffs(h,u(2)),u(3));
> simplify(CO,2);        // remove zeros among generators of CO
_[1]=x^3-z
_[2]=x^2-y
```

The equations are also obtained by computing several Sylvester resultants:

```
> resultant(f(1),f(2),t);
-x^2+y
> resultant(f(1),f(3),t);
x^3-z
```

□

6.2.2 Multipolynomial Resultants

If R is a ring and

$$F=\sum_{i=0}^{d} a_i x^i y^{d-i}\,,\quad G=\sum_{j=0}^{e} b_j x^j y^{e-j}\in R[x,y]$$

are homogeneous polynomials of positive degrees d, e in two variables x, y, we may define the resultant $\operatorname{Res}_{(d,e)}(F,G)$ of F and G using the same determinant as in the univariate case. If $R=K$ is a field, then $\operatorname{Res}_{(d,e)}(F,G)$ is zero iff the system $F=G=0$ has a **nontrivial solution** (that is, a solution in the affine space over the algebraic closure of K which is different from the origin). In fact, this essentially follows from property (R1) of the Sylvester resultant (see Cox, Little, and O'Shea (1998), Chapter 3, Proposition 1.7).

To generalize this, we consider variables $x_0,\dots,x_n$ and fix positive degrees $d_0,\dots,d_n$. If $\alpha=(\alpha_0,\dots,\alpha_n)\in\mathbb{N}^{n+1}$, we write $|\alpha|=\alpha_0+\dots+\alpha_n$ and $\boldsymbol{x}^\alpha=x_0^{\alpha_0}\cdots x_n^{\alpha_n}$. For each pair i,α, where $i\in\{0,\dots,n\}$ and $|\alpha|=d_i$, we introduce a new variable $u_{i,\alpha}$.

Theorem 6.25 (Multipolynomial Resultant). *Let $d_0,\dots,d_n$ be positive integers. There is a unique irreducible polynomial*

$$\operatorname{Res}_{(d_0,\dots,d_n)}\in\mathbb{Z}\big[u_{i,\alpha}\;\big|\;i=0,\dots,n,\;|\alpha|=d_i\big]$$

such that the following hold:

(1) If K is a field and $F_0,\dots,F_n\in K[x_0,\dots,x_n]$ are homogeneous of degrees $d_0,\dots,d_n$, then $\operatorname{Res}_{(d_0,\dots,d_n)}(F_0,\dots,F_n)$ is zero iff the system $F_0=\ldots=F_n=0$ has a nontrivial solution.
(2) $\operatorname{Res}_{(d_0,\dots,d_n)}(x_0^{d_0},\dots,x_n^{d_n})=1$.

Here, $\operatorname{Res}_{(d_0,\dots,d_n)}(F_0,\dots,F_n)\in K$ denotes the value obtained by replacing each variable $u_{i,\alpha}$ in $\operatorname{Res}_{(d_0,\dots,d_n)}$ with the coefficient of $\boldsymbol{x}^\alpha$ in F_i. □

We call the polynomial $\operatorname{Res}_{(d_0,\dots,d_n)}$ the **(multipolynomial) resultant** (in degrees $d_0,\dots,d_n$). To show the uniqueness of the resultant is an easy exercise. For a proof of existence, we refer to Gelfand, Kapranov, and Zelevinsky (1994), Chapter 13, respectively van der Waerden (1931), Chapter XI.

One important property of the resultant is that for each fixed i, it is a homogeneous polynomial in the $u_{i,\alpha}$. More precisely, we have the following result:

Theorem 6.26. *The resultant* $\mathrm{Res}_{(d_0,\dots,d_n)}$ *is a homogeneous polynomial of degree* $d_0 \cdots d_{i-1} d_{i+1} \cdots d_n$ *in the variables* $u_{i,\alpha}$, $|\alpha| = d_i$.

See Gelfand, Kapranov, and Zelevinsky (1994), Proposition 13.1.1.

Remark 6.27 (Computing Multipolynomial Resultants). By its very definition, the Sylvester resultant can be expressed as a single determinant. Except in special cases, it is not known whether this is possible for the multipolynomial resultant (see Weyman and Zelevinsky (1994) for a discussion of this problem). There are, however, several classical approaches to representing and computing the resultant (see Gelfand, Kapranov, and Zelevinsky (1994), Chapter 13 for an overview). For a recent method to find resultant formulas via exterior algebra methods, see Eisenbud and Schreyer (2003). □

The `SINGULAR` implementation of multipolynomial resultants is based on the **method of Macaulay** (1903) which we describe now. We follow the presentation in Cox, Little, and O'Shea (1998). For each i, consider the universal homogeneous polynomials of degree d_i:

$$\boldsymbol{F}_i = \sum_{|\alpha|=d_i} u_{i,\alpha} \boldsymbol{x}^\alpha \in \mathbb{Z}\big[u_{i,\alpha} \,\big|\, |\alpha| = d_i\big]\,[x_0,\dots,x_n]\,.$$

Set

$$d := -n + \sum_{i=0}^{n} d_i\,.$$

Then each $\alpha = (\alpha_0,\dots,\alpha_n) \in \mathbb{N}^{n+1}$ with $|\alpha| = d$ satisfies $\alpha_i \geq d_i$ for at least one index i (and d is the minimal integer with this property). Ordering the variables such that $x_0 > x_1 > \ldots > x_n$ and defining

$$S_i := \big\{\boldsymbol{x}^\alpha \,\big|\, |\alpha| = d \text{ and } x_i \text{ is the least variable such that } \alpha_i \geq d_i\big\}\,,$$

$i = 0,\dots,n$, we get a partition of $K[x_0,\dots,x_n]_d$ into disjoint subsets:

$$K[x_0,\dots,x_n]_d = S_0 \,\dot\cup\, \ldots \,\dot\cup\, S_n\,. \tag{6.8}$$

Consider the following system of polynomial equations:

$$\begin{array}{rl} x_0^{\alpha_0-d_0} x_1^{\alpha_1} \cdots x_n^{\alpha_n} \cdot \boldsymbol{F}_0 = 0 & \text{for all } \alpha \text{ such that } \boldsymbol{x}^\alpha \in S_0\,,\\ x_0^{\alpha_0} x_1^{\alpha_1-d_1} \cdots x_n^{\alpha_n} \cdot \boldsymbol{F}_1 = 0 & \text{for all } \alpha \text{ such that } \boldsymbol{x}^\alpha \in S_1\,,\\ \vdots \qquad\qquad & \qquad\qquad \vdots \\ x_0^{\alpha_0} x_1^{\alpha_1} \cdots x_n^{\alpha_n-d_n} \cdot \boldsymbol{F}_n = 0 & \text{for all } \alpha \text{ such that } \boldsymbol{x}^\alpha \in S_n\,. \end{array} \tag{6.9}$$

Regarding the monomials of degree d as unknowns, we get $\binom{d+n}{d}$ linear equations in $\binom{d+n}{d}$ unknowns. Let M_0 be the coefficient matrix of this system of linear equations, and set

$$D_0 := \det(M_0) \in \mathbb{Z}\big[u_{i,\alpha} \,\big|\, i = 0,\dots,n,\ |\alpha| = d_i\big]\,.$$

Then each nonzero entry of M_0 equals some $u_{i,\alpha}$ and the following holds:

Theorem 6.28. *(1) D_0 is an integer polynomial in the $u_{i,\alpha}$.*
(2) For a fixed i, D_0 is homogeneous in the $u_{i,\alpha}$ of degree equal to the number of elements in S_i. In particular, D_0 is homogeneous of degree $d_1 \cdots d_n$ in the $u_{0,\alpha}$.
(3) D_0 is divisible by the multipolynomial resultant $\operatorname{Res}_{(d_0,\ldots,d_n)}$.

For (3), note that substituting the coordinates of a zero $p \in \mathbb{A}^{\binom{d+n}{d}}$ of the resultant $\operatorname{Res}_{(d_0,\ldots,d_n)}$ for the $u_{i,\alpha}$ in the $\boldsymbol{F}_i$, the system (6.9) has a nontrivial solution (by the defining property of the resultant). Hence, the determinant D_0 vanishes at p, too.

The partition (6.8) depends on the chosen ordering of the $\boldsymbol{x}$-variables. We may repeat the construction by choosing any ordering of the variables such that x_k (in place of x_0) is the largest variable. Let D_k denote the resulting determinant. Then D_k is divisible by $\operatorname{Res}_{(d_0,\ldots,d_n)}$ and is homogeneous of degree $d_0 \cdots d_{k-1} d_{k+1} \cdots d_n$ in the $u_{k,\alpha}$. A reasoning on degrees shows:

Theorem 6.29. *The resultant $\operatorname{Res}_{(d_0,\ldots,d_n)}$ is the greatest common divisor of $D_0, \ldots, D_n$ in the ring $\mathbb{Z}\big[u_{i,\alpha} \,\big|\, i = 0, \ldots, n,\ |\alpha| = d_i\big]$ (up to sign).*

For practical applications, however, computing the polynomials $D_0, \ldots, D_n$ and their gcd is too involved. Going one step further, Macaulay represented the resultant as a quotient of two polynomials. More precisely, he showed that the "extraneous factor" $D_0 / \operatorname{Res}_{(d_0,\ldots,d_n)}$ is a minor of the matrix M_0 and he gave an explicit description of the corresponding square submatrix.

In Exercise 4.6, we will explore Macaulay's formula to compute $\operatorname{Res}_{(2,2,2)}$. We will see that this resultant (which is a polynomial in 18 variables) has total degree 12 and consists of 21894 terms. This indicates that multipolynomial resultants tend to be huge expressions which are difficult to represent and compute.

6.2.3 Zero-Dimensional Solving via Resultants

In the remaining part of this lecture, we suppose that K is a subfield of $\mathbb{C}$. We consider a (quadratic) system of equations

$$f_1(x_1, \ldots, x_n) = \ldots = f_n(x_1, \ldots, x_n) = 0\,, \tag{6.10}$$

where $f_1, \ldots, f_n \in K[x_1, \ldots, x_n]$ are polynomials of degrees $d_1, \ldots, d_n$, and we write

$$A = \mathrm{V}(f_1, \ldots, f_n) \subset \mathbb{A}^n(\mathbb{C})$$

for the set of complex solutions of the system.

In the results below, we specify conditions on $f_1, \ldots, f_n$ which allow us to find the solutions by using resultants. To be able to apply the multipolynomial resultant, we have to consider homogeneous polynomials. Homogenizing the f_i with respect to a slack variable x_0, we get n polynomials

$$F_i := f_i^{\mathrm{hom}}, \quad i = 1, \ldots, n,$$

in $n+1$ variables. Thus, we cannot apply the multipolynomial resultant directly. We overcome this problem by either adding another equation (***u*-resultant method**) or by regarding one variable as a constant as in Example 6.19 (**hidden variable method**).

The $\boldsymbol{u}$-resultant method, which is due to van der Waerden (1931), is based on the following theorem (see Cox, Little, and O'Shea (1998), Chapter 4, Proposition (2.8)). In formulating the theorem, we say that the system (6.10) has **no solutions at infinity** if the homogenized system $F_1 = \ldots = F_n = 0$ has no solutions in the hyperplane $\mathrm{V}(x_0) \subset \mathbb{P}^n(\mathbb{C})$. Equivalently, the resultant $\mathrm{Res}_{(d_1,\ldots,d_n)}(F_1|_{x_0=0}, \ldots, F_n|_{x_0=0})$ is nonzero.

Theorem 6.30 (*u*-Resultant). *Suppose that the system* (6.10) *has no solutions at infinity*[6]. *Further, let*

$$F_0 = x_0u_0 + u_1x_1 + \ldots + u_nx_n \in K(u_0, \ldots, u_n)[x_0, \ldots, x_n],$$

where $u_0, \ldots, u_n$ are auxiliary variables. Then

$$\begin{aligned}\mathrm{Res}_{(1,d_1,\ldots,d_n)}\big(F_0, F_1, \ldots, F_n\big)& \\ &= c \cdot \prod_{p\in A} (u_0 + a_1u_1 + \ldots + a_nu_n)^{\mathrm{mult}\,(p|I)},\end{aligned} \tag{6.11}$$

for some nonzero scalar $c \in K$. Here, $\mathrm{mult}\,(p \mid I)$ *denotes the multiplicity of $p = (a_1, \ldots, a_n)$ as a solution of $I = \langle f_1, \ldots, f_n\rangle$.*[7]

We call $\mathrm{Res}_{(1,d_1,\ldots,d_n)}\big(F_0, F_1, \ldots, F_n\big)$ the ***u*-resultant** of the system (6.10).

Theorem 6.30 implies that we can compute the set A by multivariate absolute polynomial factorization[8], provided we have already computed the $\boldsymbol{u}$-resultant.

Remark 6.31. It follows from Macaulay's description of the extraneous factor $D_0/\mathrm{Res}_{(d_0,\ldots,d_n)}$ that the $\boldsymbol{u}$-resultant of the system (6.10) and the determinant D_0 evaluated at the coefficients of $F_0, F_1, \ldots, F_n$ differ only by a scalar factor when regarded as polynomials in $K[u_0, \ldots, u_n]$. Moreover, for a "generic" system (6.10), the scalar is nonzero. If this is the case, the complex zeros of the system can already be read from D_0. □

Example 6.32. Using the `SINGULAR` command `mpresmat`, we compute the common zeros of a plane cubic and a plane quadric in $\mathbb{A}^2(\mathbb{C})$. In response to entering `mpresmat(I,1);`, `SINGULAR` will compute the matrix M_0 (whose determinant is D_0), evaluated at the coefficients of the given generators for `I`. Here, `SINGULAR` expects that `I` is given by $n+1$ homogeneous polynomials in

[6] Note that this condition implies that the system (6.10) is zero-dimensional.

[7] See Lecture 9, Section 9.4 for the definition of $\mathrm{mult}\,(p \mid I)$.

[8] See Lecture 7, Remark 7.1 for absolute factorization.

$n+1$ variables (the first generator must implement the $\boldsymbol{u}$-polynomial F_0). At this writing, SINGULAR returns the matrix such that the parameters $u_0, \ldots, u_n$ are replaced by the corresponding ring variables $x_0, \ldots, x_n$ (thus, factorizing $D_0 = \det(M_0)$ does not require to change the active ring):

```
> ring R = (0,u,v,w), (x,y,z), dp;
> poly f1 = x3+y-xy-1;
> poly f2 = x2+y2+4x+4y-2;
> ideal I = ux+vy+wz, homog(f1,z), homog(f2,z);
> def M0 = mpresmat(I,1);  // the evaluated matrix M_0
> nrows(M0);               // the number of rows of M_0
15
> poly D0 = det(M0);       // the value of D_0
> D0;
12*x6-64*x5y+140*x4y2-352*x3y3+804*x2y4-864*xy5+324*y6-16*x5z
+56*x4yz+32*x3y2z+48*x2y3z-336*xy4z+216*y5z-36*x4z2+64*x3yz2-
264*x2y2z2+672*xy3z2-564*y4z2+16*x3z3+32*x2yz3-208*xy2z3-
160*y3z3+20*x2z4+32*xyz4+236*y2z4-56*yz5+4*z6
```

As expected, we get a polynomial of degree 6. Applying factorize, we compute the multivariate factorization of M_0 over $\mathbb{Q}$:

```
> int aa = timer;
> factorize(D0);
[1]:
   _[1]=4
   _[2]=3*x3-x2y+9*xy2-27*y3-7*x2z-14*xyz-27*y2z+xz2+11*yz2-z3
   _[3]=x-3*y+z
   _[4]=x-y-z
   _[5]=x-y+z
[2]:
   1,1,1,1,1
> timer-aa;
0
```

We see that there are 6 complex solutions of I, namely the three rational points $(1,-3)$, $(-1,1)$, and $(1,-1)$, and a triple of points which are pairwise conjugate over $\mathbb{Q}$. For more information on the latter points, use the absFactorize command (see Lecture 7, Remark 7.1 for this command). □

From a practical point of view, multivariate polynomial factorization is usually too expensive. By conveniently specializing the coefficients u_i in F_0, it suffices to apply univariate solving. For instance, setting $\boldsymbol{u} = (u_0; -e_i)$, where e_i is the ith canonical basis vector of K^n, we get a resultant in $K[u_0]$ whose zeros are the possible x_i coordinates of the solutions.

Example 6.33. We continue the session from Example 6.32, using univariate instead of multivariate factorization:

```
> map phi = R,-1,0,z;
> poly D0x = phi(D0);
> factorize(D0x);           // x-coordinates
[1]:
   _[1]=4
   _[2]=z-1
   _[3]=z3+z2+7*z+3
   _[4]=z+1
[2]:
   1,2,1,1
> map psi = R,0,-1,z;
> poly D0y = psi(D0);
> factorize(D0y);           // y-coordinates
[1]:
   _[1]=4
   _[2]=z-1
   _[3]=z3+11*z2+27*z-27
   _[4]=z+1
   _[5]=z+3
[2]:
   1,1,1,1,1
```

Thus, the possible x values of the solutions are $1, -1, \lambda_1, \lambda_2, \lambda_3$, where $\lambda_1, \lambda_2, \lambda_3$ are the complex roots of z^3+z^2+7z+3; the possible y values are $1, -1, -3, \mu_1, \mu_2, \mu_3$, where μ_1, μ_2, μ_3 are the complex roots of $z^3+11z^2+27z-27$. □

Without printing the computation, we remark that the same sets of values are obtained by applying the hidden variable method which is based on the following theorem (see Cox, Little, and O'Shea (1998), Chapter 4, Formula (2.9)):

Theorem 6.34 (Hidden Variable). *Assume that the system* (6.10) *has no solutions at infinity. Further, let δ_i be the degree of f_i as a polynomial in $x_1, \ldots, x_{n-1}$ (with coefficients in $R = K[x_n]$), and set*

$$\widehat{F}_i = x_0^{\delta_i} \cdot f_i\left(\frac{x_1}{x_0}, \ldots, \frac{x_{n-1}}{x_0}, x_n\right) \in R[x_0, x_1, \ldots, x_{n-1}],$$

$i = 1, \ldots, n$. Then

$$\operatorname{Res}_{(\delta_1,\ldots,\delta_n)}\big(\widehat{F}_1, \ldots, \widehat{F}_n\big) = c \cdot \prod_{p \in A} (x_n - a_n)^{\operatorname{mult}(p|I)}, \tag{6.12}$$

for some nonzero scalar $c \in K$. Here, $\operatorname{mult}(p \mid I)$ *denotes the multiplicity of $p = (a_1, \ldots, a_n)$ as a solution of $I = \langle f_1, \ldots, f_n \rangle$.*

The advantage of the hidden variable method is that it involves resultants with less variables than the $\boldsymbol{u}$-resultant method. The $\boldsymbol{u}$-resultant method has the advantage that additionally to the coordinates, it provides information on how these match up to solutions (replace the u_i by different values as in Steps 3 and 4 of Algorithm 6.44 below).

Remark 6.35 (Symbolic-Numerical Approach). If one is only interested in floating point approximations of the solutions, for both methods, the univariate factorization may be replaced by a numerical univariate solver. □

As mentioned before, multipolynomial resultants tend to be huge polynomial expressions. This severely restricts their applicability to practical problems. On the other hand, most systems of polynomial equations arising from such problems are **sparse** in the sense that only a few monomials appear with a nonzero coefficient. The concept of **sparse resultants** takes this into account.

Sparse Resultants

In what follows, we roughly sketch the concept of sparse resultants. For more general statements, details and proofs, we refer to Gelfand, Kapranov, and Zelevinsky (1994), Chapter 8, and Cox, Little, and O'Shea (1998), Chapter 7. We begin by introducing some notation. A **polytope** in $\mathbb{R}^n$ is the **convex hull**

$$\operatorname{conv}(\mathcal{A}) := \left\{ \sum_{i=1}^{s} \lambda_i \alpha^{(i)} \;\middle|\; 0 \le \lambda_i \le 1\,,\ \sum_{i=1}^{s} \lambda_i = 1 \right\} \subset \mathbb{R}^n$$

of a finite subset $\mathcal{A} = \{\alpha^{(1)}, \dots, \alpha^{(s)}\}$ of $\mathbb{R}^n$. Such a polytope has **dimension** $\boldsymbol{n}$ if the vectors $\alpha^{(i)} - \alpha^{(1)}$ span $\mathbb{R}^n$ as a real vector space.

Definition 6.36. The **support** of a polynomial $f = \sum_{\alpha \in \mathbb{N}^n} c_\alpha \boldsymbol{x}^\alpha \in K[\boldsymbol{x}] = K[x_1, \dots, x_n]$ is the finite set $\mathcal{A} = \{\alpha \in \mathbb{N}^n \mid c_\alpha \neq 0\} \subset \mathbb{N}^n$. Its **Newton polytope** is the convex hull $\operatorname{conv}(\mathcal{A})$. □

Theorem 6.37 (Sparse Resultant). *Let $\mathcal{A}_0, \dots, \mathcal{A}_n$ be finite subsets of $\mathbb{N}^n$ such that each $\operatorname{conv}(\mathcal{A}_i)$ is an n-dimensional polytope. Then there is an irreducible polynomial (unique up to sign)*

$$\operatorname{Res}_{(\mathcal{A}_0, \dots, \mathcal{A}_n)} \in \mathbb{Z}\big[u_{i,\alpha} \;\big|\; i = 0, \dots, n,\ \alpha \in \mathcal{A}_i\big]$$

such that the following holds: If $g_0, \dots, g_n \in K[\boldsymbol{x}]$ are polynomials such that the system $g_0 = \ldots = g_n = 0$ has a solution in $(\mathbb{C} \setminus \{0\})^n$ and such that, for each i, the support of g_i is contained in $\mathcal{A}_i$, then $\operatorname{Res}_{(\mathcal{A}_0, \dots, \mathcal{A}_n)}(g_0, \dots, g_n) = 0$.[9]

Here, $\operatorname{Res}_{(\mathcal{A}_0, \dots, \mathcal{A}_n)}(g_0, \dots, g_n)$ denotes again the value obtained by replacing each variable $u_{i,\alpha}$ with the coefficient of $\boldsymbol{x}^\alpha$ in g_i.

Definition 6.38. The polynomial $\operatorname{Res}_{(\mathcal{A}_0, \dots, \mathcal{A}_n)}$ is referred to as a **(mixed) sparse resultant** or as the **($\boldsymbol{\mathcal{A}_0}, \dots, \boldsymbol{\mathcal{A}_n}$)-resultant**. □

Theorem 6.39. *Let $\mathcal{A}_0, \dots, \mathcal{A}_n$ be as in Theorem 6.37. Suppose, additionally, that the union of the $\mathcal{A}_i$ generates $\mathbb{Z}^n$ as an affine lattice (that is, there are finitely many $\alpha^{(j)} \in \mathcal{A}_0 \cup \ldots \cup \mathcal{A}_n$ such that $\sum_j (\alpha^{(j)} - \alpha^{(1)}) \cdot \mathbb{Z} = \mathbb{Z}^n$).*

[9] Note that it may happen that $\operatorname{Res}_{(\mathcal{A}_0, \dots, \mathcal{A}_n)}(g_0, \dots, g_n) = 0$ though the system $g_0 = \ldots = g_n = 0$ has no solution in the complex torus $(\mathbb{C} \setminus \{0\})^n$.

Then, as a polynomial in the variables $u_{i,\alpha}$, $\alpha \in \mathcal{A}_i$, the mixed sparse resultant $\mathrm{Res}_{(\mathcal{A}_0,\dots,\mathcal{A}_n)}$ *is homogeneous of degree*

$$\deg_i(\mathrm{Res}_{(\mathcal{A}_0,\dots,\mathcal{A}_n)}) = \sum_{\ell=1}^{n} (-1)^{n-\ell} \sum_{\substack{T\subset\{0,\dots,n\}\setminus\{i\}\\ |T|=\ell}} \mathrm{Vol}_n\left(\sum_{j\in T}\mathrm{conv}(\mathcal{A}_j)\right).$$

Here, Vol_n denotes the n-dimensional volume and $\sum_{j\in T}\mathrm{conv}(\mathcal{A}_j)$ refers to the **Minkowski sum**

$$\sum_{j\in T}\mathrm{conv}(\mathcal{A}_j) = \left\{ \sum_{j\in T} p_j \in \mathbb{R}^n \;\middle|\; p_j \in \mathrm{conv}(\mathcal{A}_j) \text{ for all } j \in T \right\}.$$

Remark 6.40 (Computing Sparse Resultants). To our knowledge, the first constructive method for computing and evaluating sparse resultants was proposed by Sturmfels (1993a) (for the case $\mathcal{A}_0 = \ldots = \mathcal{A}_n$). Later on, Canny and Emiris (1993-2000) designed several algorithms for computing mixed sparse resultants. A nice description of such an algorithm (partly following the lines of Macaulay's method for the computation of multipolynomial resultants) can be found in Cox, Little, and O'Shea (1998), Section 7.6. □

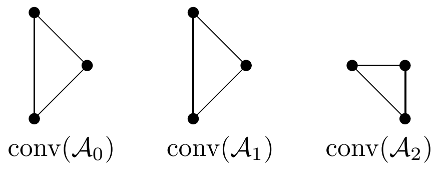
Example 6.41. Consider the three quadratic polynomials

$$f_0 = a_1 + a_2xy + a_3y^2\,,\ \ f_1 = b_1 + b_2xy + b_3y^2\,,\ \ f_2 = c_1x + c_2y + c_3xy$$

as elements of $\mathbb{Q}(\boldsymbol{a},\boldsymbol{b},\boldsymbol{c})[x,y]$. The respective supports are

$$\mathcal{A}_0 = \mathcal{A}_1 = \{(0,0),(1,1),(0,2)\}\,,\ \mathcal{A}_2 = \{(1,0),(0,1),(1,1)\}\,.$$

In particular, each convex hull $\mathrm{conv}(\mathcal{A}_i)$ is a two-dimensional polytope:



We may compute the sparse resultant $\mathrm{Res}_{\mathcal{A}_0,\mathcal{A}_1,\mathcal{A}_2}(f_0,f_1,f_2) \in \mathbb{Q}(\boldsymbol{a},\boldsymbol{b},\boldsymbol{c})$ using the `SINGULAR` command `mpresmat`. More precisely, when called with the integer `0` as a second input, `mpresmat` computes a matrix M_0 whose determinant is some multiple of the mixed sparse resultant (evaluated at the coefficients of the given generators for the first entry of `mpresmat`). The implemented algorithm is based on Canny and Emiris (2000). In the construction of the matrix M_0, combinatorial rules involving particular subdivisions of the Minkowski sum $\sum_i \mathrm{conv}(\mathcal{A}_i)$ are applied.

```
> ring R = (0,a(1..3),b(1..3),c(1..3)), (x,y), dp;
> poly f0 = a(1)+a(2)*xy+a(3)*y2;
> poly f1 = b(1)+b(2)*xy+b(3)*y2;
> poly f2 = c(1)*x+c(2)*y+c(3)*xy;
> ideal I = f0, f1, f2;
> def M0 = mpresmat(I,0);      // M0 depends on random choices
> print(M0);
(b(1)),0,     0,     0,     0,     0,     0,     0,     0,
0,     (a(1)),0,     (b(1)),0,     0,     0,     0,     0,
(b(3)),0,     (a(1)),0,     (b(1)),0,     (c(2)),0,     0,
0,     (a(3)),0,     (b(3)),0,     0,     0,     (c(2)),0,
0,     0,     (a(3)),0,     (b(3)),0,     0,     0,     0,
(b(2)),0,     0,     0,     0,     (b(1)),(c(1)),0,     (a(1)),
0,     (a(2)),0,     (b(2)),0,     0,     (c(3)),(c(1)),0,
0,     0,     (a(2)),0,     (b(2)),(b(3)),0,     (c(3)),(a(3)),
0,     0,     0,     0,     0,     (b(2)),0,     0,     (a(2))
> def p = det(M0);
> p;
(-a(1)^3*b(1)*b(2)*b(3)^2*c(3)^2+ [...]
```

Considering the determinant $\det(M_0)$ as an element of $\mathbb{Z}[\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}]$, we see that it has degree 3 as a polynomial in the a_i, degree 4 in the b_i and degree 2 in the c_i. On the other hand, Theorem 6.39 yields that the sparse resultant has degree 3 in the a_i, degree 3 in the b_i and degree 2 in the c_i. Thus, $\det(M_0)$ must have a (homogeneous) extraneous linear factor in the b_i. We determine this factor via multivariate polynomial factorization:

```
> ring S = 0, (a(1..3),b(1..3),c(1..3)), dp;
> poly p = imap(R,p);
> factorize(p,1);
_[1]=b(1)
_[2]=a(3)^3*b(1)^2*b(2)*c(1)^2-a(2)*a(3)^2*b(1)^2*b(3)*c(1)^2- [...]
```

$\square$

We come back to our polynomials $f_1, \ldots, f_n \in K[\boldsymbol{x}]$ and their set of solutions

$$A = \mathrm{V}(f_1, \ldots, f_n) \subset \mathbb{A}^n = \mathbb{A}^n(\mathbb{C}) \,.$$

Let $\mathcal{A}_i$ be the support of f_i, $i = 1, \ldots, n$. As a consequence of Theorem 1.1 in Pedersen and Sturmfels (1993), we obtain a sparse version of Theorem 6.30:

Theorem 6.42 (Sparse $\boldsymbol{u}$-Resultant). *With notations as above, suppose that, for each $1 \leq i \leq n$, the Newton polytope* $\mathrm{conv}(\mathcal{A}_i)$ *of f_i is an n-dimensional polytope. Moreover, suppose that $A \subset (\mathbb{C} \setminus \{0\})^n$. Let*

$$f_0 = u_0 + u_1 x_1 + \ldots + u_n x_n \in K(u_0, \ldots, u_n)[\boldsymbol{x}] \,,$$

where $u_0, \ldots, u_n$ are auxiliary variables, and let $\mathcal{A}_0 = \{\mathbf{0}, e_1, \ldots, e_n\}$, where e_i denotes the ith canonical basis vector of $\mathbb{R}^n$. Then

$$\begin{aligned}\operatorname{Res}_{(\mathcal{A}_0,\mathcal{A}_1,\ldots,\mathcal{A}_n)}\bigl(f_0,f_1,\ldots,f_n\bigr)& \\ &= c\cdot\prod_{p\in A}(u_0+a_1u_1+\ldots+a_nu_n)^{\operatorname{mult}(p|I)}\,,\end{aligned}\tag{6.13}$$

for some scalar $c\in K$. Here, mult $(p\mid I)$ *denotes the multiplicity of the point $p=(a_1,\ldots,a_n)$ as a solution of $I=\langle f_1,\ldots,f_n\rangle$. If the coefficients of the f_i are generically chosen*[10]*, then c is nonzero.*

Remark 6.43. The statement remains true, if we replace the sparse resultant $\operatorname{Res}_{(\mathcal{A}_0,\mathcal{A}_1,\ldots,\mathcal{A}_n)}\bigl(f_0,f_1,\ldots,f_n\bigr)$ by the determinant of the sparse resultant matrix M_0 introduced in Example 6.41 (see Canny and Emiris (2000), Theorem 7.4 and Corollary 6.5).

Even more, if each f_i, $i=1,\ldots,n$, has an n-dimensional Newton polytope, and if $\det(M_0)\neq 0$, then there is a nonzero scalar c' such that

$$\det(M_0)=c'\cdot\prod_{p\in A'}(u_0+a_1u_1+\cdots+a_nu_n)^{\operatorname{mult}(p|I)}\,,$$

for some subset $A'\subset A$ containing $A\cap(\mathbb{C}\setminus\{0\})^n$. $\square$

We, thus, have the skeleton of an algorithm for solving quadratic zero-dimensional systems of complex polynomial equations via sparse resultants:

Algorithm 6.44 (`ures_solve`).

INPUT:	$f_1,\ldots,f_n\in K[\boldsymbol{x}]=K[x_1,\ldots,x_n]$, $K\subset\mathbb{C}$ a subfield.
ASSUME:	$f_1,\ldots,f_n$ have n-dimensional Newton polytopes.
OUTPUT:	a set of complex solutions of $f_1=\ldots=f_n=0$, including all solutions which lie in the complex torus $(\mathbb{C}\setminus\{0\})^n$, or an error message (nongeneric case).

Step 1. Make $R:=K(u_0,\ldots,u_n)[x_1,\ldots,x_n]$ the active ring, map $f_1,\ldots,f_n$ to R, and set $f_0:=u_0+u_1x_1+\ldots+u_nx_n$.

Step 2. Construct $M_0(u_0;u_1,\ldots,u_n)$, the sparse resultant matrix evaluated at $(f_0,\ldots,f_n)$. If $\det(M_0(u_0;u_1,\ldots,u_n))=0$, return an error message.

Step 3. Compute a set containing all candidates for solutions: let L_i be the set of all roots of $\det\bigl(M_0(u_0;-e_i)\bigr)$. Then the ith coordinate of each solution $(a_1,\ldots,a_n)\in(\mathbb{C}\setminus\{0\})^n$ of $f_1=\ldots=f_n=0$ is contained in L_i.

Step 4. Identify the solutions among the candidates: solve successively

$$\det\bigl(M_0(u_0;n_1,\ldots,n_i,0,\ldots,0)\bigr)=0\,,$$

where $n_1,\ldots,n_i$ are random integers, and compare the zeros with the candidates $a_1n_1+\ldots+a_in_i$ as computed in Step 3.

[10] The set of all polynomial vectors $(f_1,\ldots,f_n)$ with nonvanishing scalar factor c in (6.13) is Zariski dense in the product of the spaces of polynomials with support in $\mathcal{A}_i$, $i=1,\ldots,n$.

Step 5. Return the solutions. □

Example 6.45. We reconsider Example 6.32, now computing a sparse resultant matrix:

```
> ring R = (0,u,v,w), (x,y), dp;
> poly f1 = x3+y-xy-1;
> poly f2 = x2+y2+4x+4y-2;
> ideal I = u+vx+wy, f1, f2;
> def M0 = mpresmat(I,0);  // the evaluated sparse resultant matrix
> nrows(M0);
13
> def D0 = det(M0);
> ring S = 0, (u,v,w), dp;
> poly D0 = imap(R,D0);
> map phi = S,u,-1,0;
> factorize(phi(D0));      // determine x-coordinates of solutions
[1]:
   _[1]=2
   _[2]=u-1
   _[3]=u3+u2+7u+3
   _[4]=u+1
[2]:
   1,2,1,1
> map psi = S,u,0,-1;
> factorize(psi(D0));      // determine y-coordinates of solutions
[1]:
   _[1]=2
   _[2]=u-1
   _[3]=u3+11u2+27u-27
   _[4]=u+1
   _[5]=u+3
[2]:
   1,1,1,1,1
```

From the output, we read the same possible x and y values of the solutions as in Example 6.33. We continue with Step 4 of the algorithm, choosing $n_1 = 1$, $n_2 = 3$:

```
> map psi_13 = S,u,1,3;
> factorize(psi_13(D0));
[1]:
   _[1]=2
   _[2]=u3-34u2+292u+648
   _[3]=u-2
   _[4]=u+2
   _[5]=u-8
[2]:
   1,1,1,1,1
```

The output shows, in particular, that each rational solution (a_1, a_2) of $f_1 = f_2 = 0$ satisfies $a_1 + 3a_2 \in \{-2, 2, -8\}$. Thus, we can exclude the candidates $(1, 1)$, $(-1, -1)$, and $(-1, -3)$. Moreover, we see that each solution with a nonrational x-coordinate also has a nonrational y-coordinate. A more careful analysis allows us to exclude 6 of the 9 nonrational candidates for solutions as well. $\square$

The `SINGULAR` library `solve.lib` provides the built-in command `ures_solve` implementing Algorithm 6.44. Instead of making use of univariate factorization as we did above, a numerical solver is applied in Steps 3 and 4 of the algorithm.

Remark 6.46 (Further Reading). For a broader treatment of the solving problem, we refer to Sturmfels (2002), Dickenstein and Emiris eds. (2005), and to Sommese and Wampler (2005). For more on elimination methods, see Wang (2001). To learn more about resultants, we refer to Gelfand, Kapranov, and Zelevinsky (1994), Cox, Little, and O'Shea (1998), Sturmfels (1997), and Cattani and Dickenstein (2005).

Lecture 7

Primary Decomposition and Normalization

In this lecture, we give answers to two of the computational questions posed in Lecture 2. We begin by discussing algorithms for primary decomposition. Then we study an algorithm for normalization and explain how normalization is related to primary decomposition.

7.1 Primary Decomposition

To be prepared for this section, recall the basic definitions and results on primary decomposition stated in Lecture 2, Remark 2.6. Note that for many applications of primary decomposition in algebraic geometry, it is not really necessary to compute a complete primary decomposition. Some of the information which can be extracted from a primary decomposition may be computed directly, by cheaper algorithms.

We fix some notation. Let $I \subsetneq K[\boldsymbol{x}] = K[x_1, \dots, x_n]$ be an ideal, and let d be its dimension. Then the primary components of I of dimension d have the largest dimension among all primary components of I. They are, thus, isolated and uniquely determined by I. Their intersection, written $\mathrm{E}(I)$, is called the **equidimensional hull** of I. The radical of $\mathrm{E}(I)$, written $\sqrt[\text{equi}]{I}$, is called the **equidimensional radical** of I. If I is unmixed, that is, if there are no embedded components, then all primary components of I are uniquely determined by I. In this case, for each $v \leq d$, we write $\mathrm{E}_v(I)$ for the intersection of the primary components of I of dimension v (if there are no components of dimension v, we set $\mathrm{E}_v(I) = K[\boldsymbol{x}]$). The intersection $I = \bigcap_{v \leq d} \mathrm{E}_v(I)$ is, then, called the **equidimensional decomposition** of I.

With this notation, we list some of the computational problems arising in the context of primary decomposition:

- Compute a primary decomposition of I.
- Compute all associated primes of I.
- Compute the minimal associated primes of I.

- Compute the radical $\sqrt{I}$ of I.
- Compute the equidimensional hull $\mathrm{E}(I)$.
- Compute the equidimensional radical $\sqrt[\text{equi}]{I}$.
- For each $v \leq d$, compute an ideal whose associated primes are the associated primes of I of dimension v. By abuse of notation, we say that this means to compute a **weak equidimensional decomposition** of I.
- If I is unmixed, compute the equidimensional decomposition of I.

The first algorithms for these tasks were given by Grete Hermann (1926), a student of Emmy Noether. Hermann used generic projections, that is, elimination, based on resultant techniques, to reduce to the case of hypersurfaces (algebraically, to the case of principal ideals). Thus, computing radicals reduces to the computation of the square-free part of polynomials and computing primary decomposition reduces to factorizing polynomials. In contrast to the algorithms based on Gröbner basis techniques, algorithms for polynomial factorization highly depend on the nature of the underlying field. We refer to the survey papers by Kaltofen (1982, 1990, 1992, 2003) for the history of univariate and multivariate polynomial factorization over various coefficient domains.

Remark 7.1 (The Role of the Coefficient Field). The basic `SINGULAR` command for polynomial factorization is the `factorize` command. It applies to polynomials with coefficients in a finite field, in $\mathbb{Q}$, or in a number field. For instance:

```
> ring R1 = 0, x, dp;
> poly f1 = 3x7+3x6-18x5-30x4+15x3+63x2+48x+12;
> factorize(f1);
[1]:
   _[1]=3
   _[2]=x+1
   _[3]=x-2
[2]:
   1,5,2
```

We read from the output that $f_1 = 3(x+1)^5(x-2)^2$ is the factorization of f_1 into irreducible factors in $\mathbb{Q}[\boldsymbol{x}]$. To control the output, we can add an integer as a second input of `factorize`. If this integer is `1`, for instance, the constant factor (here, `3`) and the multiplicities (here, `1,5,2`) will not be printed.

We consider further examples:

```
> factorize(x4-2,1);
_[1]=x4-2
> ring R2 = (0,a), x, dp;  minpoly = a4-2;
> factorize(x4-2,1);
_[1]=x2+(a2)
_[2]=x+(a)
_[3]=x+(-a)
```

```
> ring R3 = 0, (x,y), dp;
> poly f3 = x2y4+y6+2x3y2+2xy4-7x4+7x2y2+14y4+6x3+6xy2+47x2+47y2;
> factorize(f3,1);
_[1]=y4+2xy2-7x2+14y2+6x+47
_[2]=x2+y2
```

We see that `factorize` computes the factorization over the given coefficient field and not over a finite extension where the decomposition into the absolutely irreducible factors occurs. Here, a polynomial with coefficients in K is called **absolutely irreducible** if it is irreducible over the algebraic closure of K.

SINGULAR kernel commands for **absolute polynomial factorization** will be available with future versions of SINGULAR. At this writing, the library `absfact.lib` written by Lecerf provides the command `absFactorize` which is based on ideas of Trager (1976)[1]. The command takes as input a polynomial with coefficients in $\mathbb{Q}$, say $f \in \mathbb{Q}[x_1, \dots, x_n]$. For each irreducible factor g of f over $\mathbb{Q}$, `absFactorize` computes

- one absolutely irreducible factor of g, say g_1, and
- a minimal polynomial specifying a finite extension of $\mathbb{Q}$ of minimal degree over which g_1 is defined.

Note that the absolutely irreducible factors of g are pairwise **conjugate** over $\mathbb{Q}$, they form an orbit under the natural action of the Galois group of $\overline{\mathbb{Q}}$ over $\mathbb{Q}$ on $\overline{\mathbb{Q}}[x_1, \dots, x_n]$, where $\overline{\mathbb{Q}}$ is the algebraic closure of $\mathbb{Q}$. This does not mean, however, that all absolutely irreducible factors of g are defined over the field specified for g_1. Here is an example:

```
> LIB "absfact.lib";
> setring R1;
> def R4 = absFactorize(x4-2);
```

At this point, SINGULAR gives an explanation on how to access what has been computed. We follow the explanation (without printing it in our notes):

```
> setring R4;
> absolute_factors;
[1]:
   _[1]=1
   _[2]=x+(-a)
[2]:
   1,1
[3]:
   _[1]=(a)
   _[2]=(a4-2)
[4]:
   4
```

[1] For more details on the implemented algorithm, see Chèze and Lecerf (2005).

The list `absolute_factors` has four entries, referring to the absolutely irreducible factors (one per conjugacy class), the multiplicities, the minimal polynomials, and the total number of absolutely irreducible factors. In our example, we spot the absolutely irreducible factor $x - a$ which is defined over $\mathbb{Q}(a) = \mathbb{Q}[t]/\langle t^4 - 2\rangle$. Altogether, there are 4 absolutely irreducible factors which are pairwise conjugate over $\mathbb{Q}$. As we saw on Page 202, two of these factors are not defined over $\mathbb{Q}(a)$.

For the polynomial `f3` in the ring `R3` considered above, we obtain:

```
> setring R3;
> def R5 = absFactorize(f3);
> setring R5;
> absolute_factors;
[1]:
   _[1]=1/169
   _[2]=13*y2+(-14a+19)*x+(-7a+94)
   _[3]=x+(a)*y
[2]:
   1,1,1
[3]:
   _[1]=(a)
   _[2]=(7a2-6a-47)
   _[3]=(a2+1)
[4]:
   4
```

We see that `f3` decomposes into 2 pairs of absolutely irreducible factors such that the factors of each pair are conjugate over $\mathbb{Q}$. One pair is represented by the factor $13y^2 - 14ax + 19x - 7a + 94$, where a is a complex zero of $7t^2 - 6t - 47$. The other pair is $x + iy$, $x - iy$, where $i^2 = -1$. $\square$

We can, in particular, use `absFactorize` to check whether a given polynomial over $\mathbb{Q}$ is absolutely irreducible.

Example 7.2. We return to the polynomial $f = y^4 + z^2 - y^2(1-x^2) \in \mathbb{Q}[x, y, z]$ considered in Lecture 2, Examples 2.1 and 2.5:

```
> LIB "absfact.lib";
> ring R = 0, (x,y,z), dp;
> poly f = y4+z2-y2*(1-x2);
> def S = absFactorize(f);
> setring S;
> absolute_factors[4];  // number of absolutely irreducible factors
1
```
$\square$

Providing efficient algorithms for the problems arising in the context of primary decomposition is difficult and still one of the big challenges in computer algebra. A survey on some of the modern algorithms can be found in Decker, Greuel, and Pfister (1999). Most of the algorithms follow the basic strategy of

Hermann. They use elimination, now based on Gröbner basis techniques (and, in the case of Wang's algorithm below, in addition on characteristic set methods), to reduce to square-free decomposition or to polynomial factorization. A different approach is taken by Eisenbud, Huneke, and Vasconcelos (1992) (EHV for short), who avoid generic projections, relying on syzygy methods instead. The EHV algorithm for computing the equidimensional hull, for instance, is based on the following result:

Proposition 7.3. *If $I \subsetneq R = K[\boldsymbol{x}]$ is an ideal, then*

$$\mathrm{E}(I) = \operatorname{ann}\operatorname{Ext}_R^{n-d}(R/I, R), \text{ where } d = \dim I.$$

Here, if M is a module over a ring R, its **annihilator** is the ideal

$$\operatorname{ann} M = \{f \in R \mid fM = 0\} = 0 : M \subset R\,.$$

Algorithms which compute, possibly in combination with other algorithms, a complete primary decomposition are given in the following papers:

- Gianni, Trager, and Zacharias (1988);
- Wang (1989, 2001) (minimal associated primes only);
- Eisenbud, Huneke, and Vasconcelos (1992);
- Shimoyama and Yokoyama (1996) (suppose that the minimal associated primes are given).

Wang's algorithm finds the minimal associated primes by combining the characteristic set decomposition technique with Gröbner bases (needed for saturation). Eisenbud, Huneke and Vasconcelos use Gröbner bases to reduce primary decomposition to normalization (see Remark 7.12 at the end of this lecture).

The starting point of the algorithm of Gianni, Trager, and Zacharias (GTZ for short) is the following simple observation:

Lemma 7.4 (Splitting Tool). *If $I \subset K[\boldsymbol{x}]$ is an ideal, if $h \in K[\boldsymbol{x}]$ is a polynomial, and if $m \geq 1$ is an integer such that $I : \langle h\rangle^\infty = I : \langle h\rangle^m$, then*

$$I = \big(I : \langle h\rangle^m\big) \cap \langle I, h^m\rangle\,.$$

The key result on which the algorithm is based specifies which polynomials h are considered:

Proposition 7.5. *Let $I \subsetneq K[\boldsymbol{x}]$ be a proper ideal, and let $\boldsymbol{u} \subset \boldsymbol{x}$ be a subset of maximal cardinality such that $I \cap K[\boldsymbol{u}] = \{0\}$. Then:*

(1) The ideal $I\,K(\boldsymbol{u})[\boldsymbol{x}\setminus\boldsymbol{u}] \subset K(\boldsymbol{u})[\boldsymbol{x}\setminus\boldsymbol{u}]$ is zero-dimensional.
(2) Let $> = (>_{\boldsymbol{x}\setminus\boldsymbol{u}}, >_{\boldsymbol{u}})$ be a global product order on $K[\boldsymbol{x}]$, and let $\mathcal{G}$ be a Gröbner basis for I with respect to $>$. Then $\mathcal{G}$ is a Gröbner basis for $I\,K(\boldsymbol{u})[\boldsymbol{x}\setminus\boldsymbol{u}]$ with respect to the monomial order obtained by restricting

$>$ to the monomials in $K[\boldsymbol{x}\setminus\boldsymbol{u}]$. Further, if $h\in K[\boldsymbol{u}]$ is the least common multiple of the leading coefficients of the elements of $\mathcal{G}$ (regarded as polynomials in $K(\boldsymbol{u})[\boldsymbol{x}\setminus\boldsymbol{u}]$), then

$$I\,K(\boldsymbol{u})[\boldsymbol{x}\setminus\boldsymbol{u}]\cap K[\boldsymbol{x}] = I:\langle h\rangle^{\infty}\,.$$

(3) All primary components of the ideal $I\,K(\boldsymbol{u})[\boldsymbol{x}\setminus\boldsymbol{u}]\cap K[\boldsymbol{x}]$ have the same dimension, namely $\dim I$*. Further, if $I\,K(\boldsymbol{u})[\boldsymbol{x}\setminus\boldsymbol{u}] = Q_1\cap\ldots\cap Q_r$ is the minimal primary decomposition, then*

$$I\,K(\boldsymbol{u})[\boldsymbol{x}\setminus\boldsymbol{u}]\cap K[\boldsymbol{x}] = (Q_1\cap K[\boldsymbol{x}])\cap\ldots\cap(Q_r\cap K[\boldsymbol{x}])$$

is the minimal primary decomposition, too.

Taking Theorem 6.3 in Lecture 6 and the discussion right before that theorem into account, (1) and the first statement of (3) do not come as a great surprise. In fact, it is easy to see that if $>$ is a global monomial order on $K[\boldsymbol{x}]$, then every subset $\boldsymbol{u}\subset\boldsymbol{x}$ of maximal cardinality satisfying $\mathrm{L}_>(I)\cap K[\boldsymbol{u}]=\{0\}$ is also a subset of maximal cardinality such that $I\cap K[\boldsymbol{u}]=\{0\}$.

By recursion, the proposition allows us to reduce the general case of primary decomposition to the zero-dimensional case. In turn, if $I\subset K[\boldsymbol{x}]$ is a zero-dimensional ideal "in general position" (with respect to the lexicographic order satisfying $x_1>\cdots>x_n$), and if h_n is a generator for $I\cap K[x_n]$ (see Remark 6.4 in Lecture 6), the minimal primary decomposition of I is obtained by factorizing h_n. In characteristic zero, the condition that I is in general position can be achieved by means of a generic linear coordinate transformation (since such a transformation destroys sparseness, it should only be applied if this is really needed). See, for instance, Greuel and Pfister (2002), Chapter 4 for details.

Finally, the algorithm of Shimoyama and Yokoyama is a variant of the GTZ algorithm reducing primary decomposition in the general case to the computation of the minimal associated primes.

The GTZ algorithm and a combination of the algorithms of Wang and Shimoyama-Yokoyama are implemented in the `SINGULAR` library `primdec.lib`. The corresponding commands are `primdecGTZ` and `primdecSY`.

Remark 7.6 (The Role of the Coefficient Field). The `primdecGTZ` and `primdecSY` commands decompose an ideal over the given coefficient field. For `primdecSY`, this may be a finite field, the field $\mathbb{Q}$, or a number field. For `primdecGTZ`, recall from the discussion above that the GTZ algorithm is designed for coefficient fields of characteristic zero. In its `SINGULAR` implementation, it works over $\mathbb{Q}$ and over number fields. It also works correctly over finite fields as long as it terminates. Over a small finite field, however, it may well run into an indefinite loop (trying to find an appropriate coordinate transformation as discussed above).

The library `primdec.lib` also provides the `absPrimdecGTZ` command which relies on the `absFactorize` command. We explain its usage towards the end of the following example. □

Note that `primdec.lib` provides, in fact, implementations of algorithms for all tasks listed at the beginning of this section. For each task, no "generally best" algorithm is known. Depending on the example, the difference in the performance of the various algorithms could mean to get a result, or to run out of time or memory.

Example 7.7 (Butcher). We show some commands of `primdec.lib` at work.

```
> ring R = 0, (a,b,c,d,e,f,g,h), dp;
> ideal I = a+c+d-e-h, 2df+2cg+2eh-2h2-h-1,
.           3df2+3cg2-3eh2+3h3+3h2-e+4h,
.           6bdg-6eh2+6h3-3eh+6h2-e+4h,
.           4df3+4cg3+4eh3-4h4-6h3+4eh-10h2-h-1,
.           8bdfg+8eh3-8h4+4eh2-12h3+4eh-14h2-3h-1,
.           12bdg2+12eh3-12h4+12eh2-18h3+8eh-14h2-h-1,
.           -24eh3+24h4-24eh2+36h3-8eh+26h2+7h+1;
```

We compute a primary decomposition of `I`:

```
> LIB "primdec.lib";
> int aa = timer;
> list PD = primdecGTZ(I);
> timer-aa;     // time in seconds
59
> size(PD);
9
```

Thus, there are 9 components. For each component, `SINGULAR` stores the primary ideal and the corresponding associated prime. In our example, one can check that there are three components of dimension 3, four components of dimension 2, and two components of dimension 0. For instance:

```
> ideal Prime6 = std(PD[6][2]);
> dim(Prime6);
2
```

Thus, the 6th component has dimension 2. We print this component:

```
> PD[6];
[1]:
   _[1]=h2+2h+1
   _[2]=gh+g
           [...]
   _[15]=-144h5-408h4-384h3-118h2+8e+3h+9
   _[16]=-144h5-408h4-384h3-118h2+8a+8c+8d-5h+9
[2]:
   _[1]=h+1
   _[2]=g
   _[3]=f
   _[4]=d
   _[5]=-144h5-408h4-384h3-118h2+8e+3h+9
   _[6]=-144h5-408h4-384h3-118h2+8a+8c+8d-5h+9
```

Let us check that the ideal PD[6][2] is indeed the radical of PD[6][1]. In SINGULAR, the command radical is based on the algorithm of Kemper (2002) in positive characteristic and on the GTZ type algorithm of Krick and Logar (1991) in characteristic 0 (this algorithm makes use of Proposition 7.5 to reduce the general case to the zero-dimensional case which can be settled as explained in Lecture 6, Remark 6.6). The command radicalEHV is based on the ideas of Eisenbud, Huneke and Vasconcelos (we do not recommend to apply this command for the ideal under consideration).

```
> ideal check = std(radical(PD[6][1]));
> check;
check[1]=h+1
check[2]=g
check[3]=f
check[4]=e+1
check[5]=d
check[6]=a+c+2
> size(reduce(Prime6,check,1));
0
> size(reduce(check,Prime6,1));
0
```

The equidimensional hull can also be computed by a GTZ type algorithm. The corresponding SINGULAR command is equidimMax (in our example, because of the high codimension of I, we do not use the command equidimMaxEHV based on Proposition 7.3).

```
> ideal EI = equidimMax(I);
> dim(std(EI));
3
> list PDEI = primdecGTZ(EI);
> size(PDEI);
3
```

This shows that there are 3 components of top dimension 3.

Finally, we apply absPrimdecGTZ:

```
> aa = timer;
> def S = absPrimdecGTZ(I);
> timer-aa;
123
```

The return value of absPrimdecGTZ(I) is a ring S which comes with a list absolute_primes. This list has the same number of entries as the list returned by primdecGTZ(I). The entry absolute_primes[j] has as first entry an ideal describing one representative of the class of pairwise **conjugate absolute primes** corresponding to the jth prime ideal PD[j] returned by primdecGTZ(I) (here, we use a notation similar to that in Remark 7.1). The second entry refers to the number of conjugates in the class.

In our example, PD[6][2] is an absolute prime:

```
> setring S;
> absolute_primes[6];
[1]:
   _[1]=@c
   _[2]=h+1
        [...]
   _[7]=a+c+2
[2]:
   1
```

In contrast, PD[9][2] decomposes into 2 absolute primes:

```
> absolute_primes[9];
[1]:
   _[1]=12*@c^2+12*@c+1
   _[2]=h-@c
   _[3]=2*g+2*@c+1
   _[4]=f+2*@c+1
   _[5]=e+1
   _[6]=4*d+@c+1
   _[7]=2*c+@c+1
   _[8]=b+2*@c+1
   _[9]=4*a-7*@c+1
[2]:
   2
```

The first generator12*@c^2+12*@c+1 of absolute_primes[9] refers to the minimal polynomial of a finite extension of $\mathbb{Q}$ (of minimal degree) over which the absolute prime is defined.

The total number of absolute primes is obtained as follows:

```
> int num_abs_primes;
> for (int j=1; j<=size(absolute_primes); j++)
.   { num_abs_primes = num_abs_primes + absolute_primes[j][2]; }
> num_abs_primes;
12
```

□

Remark 7.8. (1) To compute minimal associated primes over the given coefficient field, use minAssGTZ (based on a GTZ type algorithm), or minAssChar (based on Wang's characteristic set algorithm).

(2) The SINGULAR command equidim relies on an algorithm which combines equidimMax and sat. It computes a weak equidimensional decomposition over the given coefficient field. For instance:

```
> LIB "primdec.lib";
> ring R = 0, (x,y), dp;
> ideal I = xy, y2;        // ideal is mixed
> list LI = equidim(I);
> LI;
```

```
[1]:
   _[1]=y
   _[2]=x
[2]:
   _[1]=y
> intersect(LI[1],LI[2]);
_[1]=y
```

If the ideal under consideration is unmixed, `equidim` returns an equidimensional decomposition:

```
> ideal J = xy, y3-2y2+y;
> list LJ = equidim(J);
> LJ;
[1]:
   _[1]=x
   _[2]=y2-2y+1
[2]:
   _[1]=y
> intersect(LJ[1],LJ[2]);
_[1]=xy
_[2]=y3-2y2+y
```
□

7.2 Normalization

If S is an integral domain, its **normalization** $\overline{S}$ is the integral closure of S in the quotient field of S. An important finiteness result of Emmy Noether tells us that if S is an affine domain, then $\overline{S}$ is a finitely generated S-module; in particular, $\overline{S}$ is again an affine domain (see Eisenbud (1995), Corollary 13.13). In other words, if S is of type $S = K[x_1,\dots,x_n]/P$, where P is a prime ideal, then $\overline{S}$ is of type $K[y_1,\dots,y_m]/P'$, where P' is a prime ideal. To compute the normalization means to find such a representation for $\overline{S}$ together with the normalization map

$$S = K[x_1,\dots,x_n]/P \hookrightarrow \overline{S} = K[y_1,\dots,y_m]/P'.$$

More generally, if S is any reduced ring, its **normalization** $\overline{S}$ is the integral closure of S in the total quotient ring of S. In conjunction with Noether's finiteness result, the theorem on the splitting of normalization tells us that if S is a reduced affine ring, then $\overline{S}$ may be written as a product of affine domains. More precisely, if S is of type $S = K[x_1,\dots,x_n]/I$, where I is a radical ideal, and if $P_1,\dots,P_s$ are the (minimal) associated primes of I, then

$$\overline{S} \cong \overline{(K[x_1,\dots,x_n]/P_1)} \times \cdots \times \overline{(K[x_1,\dots,x_n]/P_s)}\,.$$

See de Jong and Pfister (2000), Theorem 1.5.20.

Algorithms to compute the normalization were given by several authors. The algorithm proposed by de Jong (1998) is based on a criterion for normality due to Grauert and Remmert (1971):

Theorem 7.9. *Let S be a reduced Noetherian ring, let*

$$\mathrm{Spec}(S) = \{P \subset S \mid P \textit{ is a prime ideal}\}$$

be its **spectrum**, *let J be a radical ideal of S, and let*

$$\mathrm{V}(J) = \{P \in \mathrm{Spec}(S) \mid J \subset P\}$$

be the **vanishing locus** *of J in* $\mathrm{Spec}(S)$. *Suppose that J contains a nonzerodivisor and that* $\mathrm{V}(J)$ *contains the* **nonnormal locus**

$$\{P \in \mathrm{Spec}(S) \mid \textit{the localization } S_P \textit{ is not normal}\}$$

of $\mathrm{Spec}(S)$. *Then S is normal iff S equals the ring* $\mathrm{Hom}_S(J, J)$ *of endomorphisms of J.*

Notice that in the situation of the theorem, we have canonical inclusions of rings:

$$S \subset \mathrm{Hom}_S(J, J) \subset \overline{S}.$$

The first inclusion is the map which sends an element of S to multiplication with this element. For the second inclusion, we pick a nonzerodivisor $f \in J$ and map $\phi \in \mathrm{Hom}_S(J, J)$ to $\frac{\phi(f)}{f}$. It is easily checked that this map is independent of the choice of f. That we in fact land in $\overline{S}$ is a consequence of the theorem of Cayley-Hamilton (see Eisenbud (1995), Theorem 4.3). To compute the normalization, de Jong's algorithm proceeds as follows. If S is given, find an ideal J satisfying the assumptions of the theorem. Check whether $S = \mathrm{Hom}_S(J, J)$. If so, stop. Otherwise, replace S by the strictly larger ring $\mathrm{Hom}_S(J, J)$ and start over again. This process must eventually terminate due to Emmy Noether's finiteness result.

De Jong's algorithm is implemented in the `SINGULAR` library `normal.lib` (see Decker et al (1999)). The corresponding command `normal` takes as input a radical ideal I in a polynomial ring $R = K[x_1, \ldots, x_n]$ (if I is not radical, apply `I = radical(I);` first). The algorithm finds an ideal $J \subset S = K[x_1, \ldots, x_n]/I$ as in Theorem 7.9 as follows. The **singular locus**

$$\{P \in \mathrm{Spec}(S) \mid \text{the localization } S_P \text{ is not regular}\}$$

of $\mathrm{Spec}(S)$ contains the nonnormal locus (see for instance Greuel and Pfister (2002), Section 5.7). Thus, if K is a perfect field (for instance, a field of characteristic zero), we may apply the algebraic version of the Jacobian Criterion (see Eisenbud (1995), Section 16.6): if I is known to have pure codimension c, the algorithm picks J to be the ideal generated by the residue classes of the $c \times c$ minors of I. Otherwise, it computes an equidimensional decomposition of I first. The algorithm returns polynomial rings $R_1, \ldots, R_t$, ideals $I_j \subset R_j$, and ring maps $\pi_j : R \to R_j$ such that the induced map

$$\pi : R/I \longrightarrow \prod_{j=1}^{t} R_j/I_j, \quad f + I \longmapsto \big(\pi_1(f) + I_1, \dots, \pi_t(f) + I_t\big),$$

is the normalization map. At this writing, the algorithm is implemented so that its basic version does not necessarily return prime ideals I_j. That is, the prime components of I are not necessarily separated.

Example 7.10. We use SINGULAR to compute a normalization of the affine ring $Q[x, y, z]/I$, where $I = \langle z - x^4, z - y^6\rangle$:

```
> LIB "normal.lib";
> ring R = 0, (x,y,z), dp;
> ideal I = z-x4, z-y6;
> list nor = normal(I);
// 'normal' created a list of 1 ring(s)
```

At this point, SINGULAR gives an explanation on how to access what has been computed. We follow the explanation (without printing it in our notes). Making use of the notation introduced above, we may say that `normal` creates the list of rings $R_1, \dots, R_t$. Each ring R_j comes with two ideals referred to as `norid` and `normap`. Here, `norid` is the ideal I_j, and `normap` defines the ring map $\pi_j : R \to R_j$. In our example here, only one ring has been created. We make this ring the active ring and print `norid` and `normap`:

```
> def R1 = nor[1];  setring R1;  R1;
//   characteristic : 0
//   number of vars : 2
//        block   1 : ordering a
//                  : names    T(1) T(2)
//                  : weights     1    0
//        block   2 : ordering dp
//                  : names    T(1) T(2)
//        block   3 : ordering C
> norid;
norid[1]=T(2)^2-1
> normap;
normap[1]=T(1)^3*T(2)^2
normap[2]=T(1)^2*T(2)
normap[3]=T(1)^12*T(2)^8
```

Though `normal` computed only one ring, the ideal I has more than one primary component:

```
> setring R;
> primdecGTZ(I);
[1]:
   [1]:
      _[1]=y6-z
      _[2]=-y3+x2
      [2]:
         _[1]=y6-z
         _[2]=-y3+x2
   [2]:
      [1]:
         _[1]=y6-z
         _[2]=y3+x2
      [2]:
         _[1]=y6-z
         _[2]=y3+x2
```

We see that I is the intersection of two prime ideals. In particular, it is radical. Using another version of the normalization algorithm, the prime components are separated:

```
> list NOR = normal(I,1);
// 'normal' created a list of 2 ring(s).
```

Again, we follow the explanation printed by SINGULAR.

```
> R1 = NOR[1];          // the 1st ring
> setring R1; R1;
//   characteristic : 0
//   number of vars : 1
//        block   1 : ordering dp
//                  : names    T(1)
//        block   2 : ordering C
> norid;
norid[1]=0
> normap;
normap[1]=T(1)^3
normap[2]=-T(1)^2
normap[3]=T(1)^12
> def R2 = NOR[2];      // the 2nd ring
> setring R2; R2;
//   characteristic : 0
//   number of vars : 1
//        block   1 : ordering dp
//                  : names    T(1)
//        block   2 : ordering C
> norid;
norid[1]=0
> normap;
normap[1]=T(1)^3
normap[2]=T(1)^2
normap[3]=T(1)^12
```
□

Example 7.11. Again, we compute the normalization of an affine ring which is defined by two polynomials in three indeterminates:

```
> LIB "normal.lib";
> ring R = 0, x(1..3), dp;
> poly f1 = -x(1)^3*x(2)^3*x(3)^2-x(1)^3*x(2)*x(3)^4
.           -x(1)*x(2)^3*x(3)^4+x(1)^5*x(2)^2+x(1)^5*x(3)^2
.           +x(1)^3*x(2)^2*x(3)^2+x(1)^2*x(2)^2*x(3)^3
.           -x(1)^4*x(2)*x(3);
> poly f2 = -x(1)^2*x(2)^3*x(3)^5-x(1)^2*x(2)*x(3)^7-x(2)^3*x(3)^7
.           +x(1)^2*x(2)^5*x(3)^2+x(1)^4*x(2)^2*x(3)^3+x(2)^5*x(3)^4
.           +x(1)^4*x(3)^5+x(1)^2*x(2)^2*x(3)^5-x(1)^2*x(2)*x(3)^6
.           +x(1)*x(2)^2*x(3)^6-x(2)^3*x(3)^6-x(1)^4*x(2)^4
.           -x(1)^2*x(2)^4*x(3)^2-x(1)*x(2)^4*x(3)^3+x(1)^4*x(3)^4
.           -x(1)^3*x(2)*x(3)^4+x(1)^2*x(2)^2*x(3)^4
.           +x(1)*x(2)^2*x(3)^5+x(1)^3*x(2)^3*x(3)
.           -x(1)^3*x(2)*x(3)^3;
> ideal I = f1, f2;
> list NOR = normal(I);
// 'normal' created a list of 3 ring(s).
```

To access what has been computed, we follow the explanation given by SINGULAR:

```
> def R1 = NOR[1];      // the 1st ring
> setring R1;
> norid;
norid[1]=T(3)^2+T(4)^2+T(4)*T(5)-T(4)
norid[2]=T(2)*T(3)+T(1)*T(4)+T(1)*T(5)-T(1)
norid[3]=T(1)*T(3)-T(2)*T(4)
norid[4]=T(2)^2+T(4)*T(5)+T(5)^2-T(5)
norid[5]=T(1)*T(2)-T(3)*T(5)
norid[6]=T(1)^2-T(4)*T(5)
> normap;
normap[1]=T(1)
normap[2]=T(2)
normap[3]=T(3)
> def R2 = NOR[2];      // the 2nd ring
> setring R2;  norid;
norid[1]=0
> normap;
normap[1]=T(1)*T(2)
normap[2]=T(2)^2
normap[3]=T(1)
> def R3 = NOR[3];      // the 3rd ring
> setring R3;  norid;
norid[1]=0
> normap;
normap[1]=0
normap[2]=T(1)^3-T(1)
normap[3]=T(1)^2-1
```

Thus, the normalization has three components, an affine part of the Veronese surface in $\mathbb{P}^5$, a plane, and a line. A check on the primary decomposition of `I` shows that this is not a great surprise:

```
> setring R;
> primdecGTZ(I);
[1]:
   [1]:
      _[1]=x(1)^2*x(2)^2+x(1)^2*x(3)^2+x(2)^2*x(3)^2-x(1)*x(2)*x(3)
   [2]:
      _[1]=x(1)^2*x(2)^2+x(1)^2*x(3)^2+x(2)^2*x(3)^2-x(1)*x(2)*x(3)
[2]:
   [1]:
      _[1]=-x(2)*x(3)^2+x(1)^2
   [2]:
      _[1]=-x(2)*x(3)^2+x(1)^2
[3]:
   [1]:
      _[1]=-x(3)^3+x(2)^2-x(3)^2
      _[2]=x(1)
   [2]:
      _[1]=-x(3)^3+x(2)^2-x(3)^2
      _[2]=x(1)
```

We see that `I` is the intersection of three prime ideals, defining the Steiner Roman surface (see Exercise 1.5 (e)), the Whitney umbrella (see Lecture 9, Example 9.31), and a nodal plane curve, respectively. In particular, `I` is a radical ideal. □

Remark 7.12. Here is how Eisenbud, Huneke, and Vasconcelos (1992) reduce primary decomposition to normalization. By localization techniques, they reduce to computing the associated primes of an ideal $I \subset K[x_1, \ldots, x_n]$. For this, they may assume that I is equidimensional and radical (making use of algorithms for weak equidimensional decomposition and equidimensional radical). In addition, I can be assumed to be homogeneous. It, thus, remains to find the minimal prime ideals of the graded, reduced ring $S = K[x_1, \ldots, x_n]/I$. These can be obtained by intersecting the minimal primes of the normalization $\overline{S}$ of S with S. The latter ideals are in one-to-one correspondence with the idempotents of $\overline{S}$ and can therefore be computed. □

Remark 7.13 (Further Reading). For definitions and basic results on primary decomposition and normalization, we refer to Atiyah and MacDonald (1969), Eisenbud (1995), and Greuel and Pfister (2002). For more on the algorithms, see Decker, Greuel, and Pfister (1999).

Practical Session IV

Exercise 4.1. Compute a primary decomposition of the ideal of $\mathbb{Q}[t,w,x,y,z]$ which is generated by the polynomials below:

$$\begin{aligned}&w^2xy + w^2xz + w^2z^2,\\ &tx^2y + x^2yz + x^2z^2,\\ &twy^2 + ty^2z + y^2z^2,\\ &t^2wx + t^2wz + t^2z^2.\end{aligned}$$

How many components of which dimension do you get? Are there nonprime components? Which components are embedded?

Exercise 4.2. Compute the normalization of the affine ring

$$R = \mathbb{Q}[b,s,t,u,v,w,x,y,z]/I\,,$$

where I is the ideal

$$I := \langle wy - vz,\ vx - uy,\ tv - sw,\ su - bv,\ tuy - bvz\rangle\,.$$

What do you get?

Exercise 4.3. Compute a nonnormal affine ring of dimension 2 whose normalization is the affine ring $\mathbb{F}_{32003}\,[x,y,z]/\langle z^2 - x^5 - y^5\rangle$.

Hint. For instance, consider the subalgebra of $\mathbb{F}_{32003}\,[x,y,z]/\langle z^2 - x^5 - y^5\rangle$ generated by the residue classes $\overline{x}, \overline{z}, \overline{xy}, \overline{y}^2, \overline{yz}, \overline{y}^3$.

Exercise 4.4. Consider the matrix

$$D = \begin{pmatrix} x_1 & x_2 & x_3^2 - 1\\ x_2 & x_3 & x_1x_2 + x_3 + 1\\ x_3^2 - 1 & x_1x_2 + x_3 + 1 & 0\end{pmatrix}$$

and the ideal $I = \langle f_1, f_2\rangle \subset \mathbb{Q}[x_1,x_2,x_3]$ generated by the determinant $f_1 = \det D$ and the "first" 2×2 minor $f_2 = x_1x_3 - x_2^2$ of D. Use `SINGULAR` to verify the following statements:

(a) The ideal I has pure codimension 2 and is unmixed.

(b) The vanishing locus of the ideal $J = I_2(\frac{\partial f_i}{\partial x_j}) + I$ coincides with that of I, that is, $\mathrm{V}(J) = \mathrm{V}(I) =: A \subset \mathbb{A}^3(\mathbb{C})$.

(c) A is smooth.

Exercise 4.5. Write a `SINGULAR` procedure

- which assumes that the active ring is a univariate polynomial ring with coefficients in a prime field,
- which takes as input a polynomial f in the active ring, and
- which returns a `ring` in which f decomposes into linear factors, making the solutions of $f = 0$ accessible for later use.

Test your procedure with the polynomials

(a) $f = (x^2 + 2)(x^2 + 3)(x^2 + 5) \in \mathbb{Q}[x]$,
(b) $f = x^6 - 13 \in \mathbb{Q}[x]$, and
(c) $f = x^{100} - 13 \in \mathbb{F}_{167}[x]$.

Exercise 4.6. Compute the multipolynomial resultant $\mathrm{Res}_{2,2,2}$, and display its (total) degree and its number of terms (do not print $\mathrm{Res}_{2,2,2}$ itself). Use Macaulay's method: if

$$\begin{aligned}
\boldsymbol{F}_0 &= a_1z^2 + a_2yz + a_3y^2 + a_4xz + a_5xy + a_6x^2\,,\\
\boldsymbol{F}_1 &= b_1z^2 + b_2yz + b_3y^2 + b_4xz + b_5xy + b_6x^2\,,\\
\boldsymbol{F}_2 &= c_1z^2 + c_2yz + c_3y^2 + c_4xz + c_5xy + c_6x^2\,,
\end{aligned}$$

then, up to a sign,

$$\mathrm{Res}_{2,2,2} = \frac{D_0}{D_0^{\mathrm{ext}}} \in \mathbb{Z}[a_1, \ldots, a_6, b_1, \ldots, b_6, c_1, \ldots, c_6]\,.$$

Here, D_0 is the determinant of the matrix of coefficients of the following system of 15 equations (considered as linear equations in the 15 monomials in x, y, z of degree 4):

$$\begin{array}{l}
x^2\boldsymbol{F}_0 = xy\boldsymbol{F}_0 = xz\boldsymbol{F}_0 = \phantom{y^2\boldsymbol{F}_0 =}\ yz\boldsymbol{F}_0 \phantom{= z^2\boldsymbol{F}_2} = 0\,,\\
x^2\boldsymbol{F}_1 = xy\boldsymbol{F}_1 = xz\boldsymbol{F}_1 = y^2\boldsymbol{F}_1 = yz\boldsymbol{F}_1 \phantom{= z^2\boldsymbol{F}_2} = 0\,,\\
x^2\boldsymbol{F}_2 = xy\boldsymbol{F}_2 = xz\boldsymbol{F}_2 = y^2\boldsymbol{F}_2 = yz\boldsymbol{F}_2 = z^2\boldsymbol{F}_2 = 0\,.
\end{array}$$

Further, D_0^{ext} is the determinant of the 3×3 matrix whose entries are the coefficients of the monomials x^2y^2, x^2z^2, y^2z^2 in $x^2\boldsymbol{F}_1$, $x^2\boldsymbol{F}_2$, $y^2\boldsymbol{F}_2$.

Exercise 4.7. As in Exercise 2.3, consider a nonzero polynomial

$$\begin{aligned}
F_{\boldsymbol{a}} = a_1x^3 + a_2x^2y + a_3xy^2 + a_4y^3 + a_5x^2z + a_6xyz\\
+a_7y^2z + a_8xz^2 + a_9yz^2 + a_{10}z^3\,,
\end{aligned}$$

where $\boldsymbol{a} = (a_1, \ldots, a_{10}) \in \mathbb{C}^{10}$. Using the multivariate resultant $\mathrm{Res}_{2,2,2}$ from Exercise 4.6, find a necessary and sufficient condition in terms of $\boldsymbol{a}$ for the first partial derivatives of $F_{\boldsymbol{a}}$ to have a common zero in $\mathbb{P}^2(\mathbb{C})$. Compare your result with the result obtained in Exercise 2.3.

Lecture 8

Algorithms for Invariant Theory

As all the roads lead to Rome so I find in my own case at least that all algebraic inquiries, sooner or later, end at the Capitol of modern algebra over whose shining portal is inscribed the Theory Of Invariants.

J.J. Sylvester (1864), p. 380.

In this lecture, we return to the historical origin of Gröbner bases as presented in Lecture 1, giving an overview on recent algorithms for invariant theory. We begin by introducing the basic setting of invariant theory. Let G be a group, and let $K[\boldsymbol{x}] = K[x_1, \ldots, x_n]$ be the polynomial ring with its natural grading $K[\boldsymbol{x}] = \bigoplus_{d \geq 0} K[\boldsymbol{x}]_d$ by degree. Suppose that G is represented as a group of K-linear transformations of the K-vector space $W = K[\boldsymbol{x}]_1$ of linear forms in $K[\boldsymbol{x}]$. That is, G acts on W by means of a group homomorphism $G \to \mathrm{GL}_n(K)$, where we regard $\mathrm{GL}_n(K)$ as the group of invertible K-linear transformations of W by means of the basis $x_1, \ldots, x_n$. We extend the action of G to all of $K[\boldsymbol{x}]$ by setting, for $\pi \in G$ and $f \in K[\boldsymbol{x}]$,

$$(\pi f)(x_1, \ldots, x_n) = f(\pi^{-1} x_1, \ldots, \pi^{-1} x_n).$$

An **invariant** of G is a fixed point of this action. The set of all invariants, written

$$K[\boldsymbol{x}]^G = \{f \in K[\boldsymbol{x}] \mid \pi f = f \text{ for all } \pi \in G\},$$

is a graded subalgebra of $K[\boldsymbol{x}]$. It is called the **ring of invariants** of G (with respect to the given representation).

Example 8.1. The symmetric group S_n acts on $K[\boldsymbol{x}]$ by permuting the variables. The ring of invariants $K[\boldsymbol{x}]^{S_n}$ is the ring of symmetric polynomials. It is generated by the elementary symmetric polynomials

$$\begin{aligned} \sigma_1(x_1, \ldots, x_n) &:= x_1 + \cdots + x_n, \\ \sigma_2(x_1, \ldots, x_n) &:= x_1 x_2 + x_1 x_3 + \cdots + x_{n-1} x_n, \\ &\vdots \\ \sigma_n(x_1, \ldots, x_n) &:= x_1 \cdots x_n. \end{aligned}$$

That is, every symmetric polynomial can be represented as a polynomial in the elementary symmetric polynomials. In fact, such a representation is unique since the elementary symmetric polynomials are algebraically independent over K (there are no K-algebra relations among them). In sum,

$$K[\boldsymbol{x}]^{S_n} = K[\sigma_1, \ldots, \sigma_n]$$

is a finitely generated polynomial algebra. □

The **first fundamental problem of invariant theory** is to decide, whether a given ring of invariants $K[\boldsymbol{x}]^G$ is finitely generated as a K-algebra, and, if so, to compute a finite set of generators. The **second fundamental problem of invariant theory** is to compute the K-algebra relations on the generators. Elimination is a (costly) solution to the second problem (see Proposition 2.31 and Section 3.6.3). Example 8.10 will show how to carry this out in `SINGULAR`. Now, we focus on the first problem.

8.1 Finite Groups

In the setting introduced above, let G be a finite group. Emmy Noether (1926) showed that in this case $K[\boldsymbol{x}]^G \subset K[\boldsymbol{x}]$ is an integral ring extension. In particular, $\dim K[\boldsymbol{x}]^G = \dim K[\boldsymbol{x}] = n$. From this, Noether deduced with a non-constructive proof that $K[\boldsymbol{x}]^G$ is finitely generated.

How can we compute a finite set of generators? Since we already know that $K[\boldsymbol{x}]^G$ has dimension n, one basic idea is to proceed in two steps as follows. In the first step, we compute homogeneous invariants $p_1, \ldots, p_n$, referred to as **primary invariants**, such that $K[p_1, \ldots, p_n] \subset K[\boldsymbol{x}]^G$ is a graded Noether normalization (see, for instance, Eisenbud (1995), Chapter 13 for Noether normalization). In the second step, we compute a minimal system of homogeneous invariants $s_0 = 1, s_1, \ldots, s_m$, referred to as **secondary invariants**, such that $K[\boldsymbol{x}]^G$ is generated by $s_0, \ldots, s_m$ as a $K[p_1, \ldots, p_n]$-module. The p_i and s_j together form a (non-minimal) set of generators for the K-algebra $K[\boldsymbol{x}]^G$. To describe algorithms for computing primary and secondary invariants, we suppose that G is explicitly given as a matrix group $G \subset \mathrm{GL}_n(K)$ by a (finite) set of generating matrices. We distinguish two cases. We say that we are in the **nonmodular case** if the group order $|G|$ is invertible in K. That is, the characteristic of K is either zero or a prime not dividing $|G|$. Otherwise, we say that we are in the **modular case**.

8.1.1 The Nonmodular Case

In the nonmodular case, the map

$$\mathcal{R} : K[\boldsymbol{x}] \to K[\boldsymbol{x}], \quad f \mapsto \frac{1}{|G|} \sum_{\pi \in G} \pi f,$$

is well-defined. It is K-linear and graded of degree zero, and it projects $K[\boldsymbol{x}]$ onto $K[\boldsymbol{x}]^G$ such that $\mathcal{R} \circ \mathcal{R} = \mathcal{R}$. Moreover, it is a $K[\boldsymbol{x}]^G$-module homomorphism. We refer to it as the **Reynolds operator**.

Further, in the nonmodular case, we can precompute the Hilbert series of $K[\boldsymbol{x}]^G$ due to a result of Molien (1897). In fact, the Hilbert series equals what is nowadays called the **Molien series** of G. If $\operatorname{char} K = 0$, the Molien series is the formal power series in t obtained by expanding the rational function

$$\frac{1}{|G|} \sum_{\pi \in G} \frac{1}{\det(1 - t\pi)}$$

(see, for instance, Decker and de Jong (1998) for the Molien series in positive characteristic). Thus, for each given degree d, the dimension of $K[\boldsymbol{x}]_d^G$ is known to us a priori. Hence, we can compute a K-basis for $K[\boldsymbol{x}]_d^G$ by applying $\mathcal{R}$ to the elements of a (monomial) K-basis for $K[\boldsymbol{x}]_d$ until the correct number of linearly independent invariants has been found.

One way of computing primary invariants is to proceed degree by degree, and, if $p_1, \dots, p_{i-1}$ have already been constructed, to search for a homogeneous invariant p_i satisfying $\dim\langle p_1, \dots, p_i \rangle = n - i$ (see Decker, Heydtmann, and Schreyer (1998); an alternative method is due to Kemper (1999)). Indeed:

Lemma 8.2. *Let $p_1, \dots, p_i \in K[\boldsymbol{x}]^G$ be homogeneous. Then*

$$\dim(K[\boldsymbol{x}]^G / \langle p_1, \dots, p_i \rangle) \geq n - i,$$

with equality iff $p_1, \dots, p_i$ can be extended to a system of primary invariants for $K[\boldsymbol{x}]^G$.

Algorithm 8.3 (Primary Invariants).

INPUT:	a set Δ of generators for G.
OUTPUT:	a system of primary invariants for $K[\boldsymbol{x}]^G$.

Step 1. Initialization:

$d := 0; \; i := 0;$

Step 2. Iteration (degree by degree):

```
repeat
  • $d := d + 1$;
  • compute a $K$-basis $b_1, \dots, b_{c_d}$ for $K[\boldsymbol{x}]_d^G$;
  • compute $m = \dim\langle p_1, \dots, p_i, b_1, \dots, b_{c_d} \rangle$;
  • while $n - i > m$ do
      if $K$ is infinite then
        – $i := i + 1$;
        – find a $K$–linear combination $p_i$ of $b_1, \dots, b_{c_d}$ such that
          $\dim\langle p_1, \dots, p_{i-1} \rangle > \dim\langle p_1, \dots, p_i \rangle$;
```

```
            else
              – find a new $K$–linear combination $q$ of $b_1, \ldots, b_{c_d}$;
              – if $\dim\langle p_1, \ldots, p_i\rangle > \dim\langle p_1, \ldots, p_i, q\rangle$ then
                  $i := i + 1$, $p_i := q$;
              – if all combinations have been tested then
                  break [of while–loop]
    until $i = n$;
return$(p_1, \ldots, p_n)$.
```
□

The computation of secondary invariants is based on the fact that, in the nonmodular case, $K[\boldsymbol{x}]^G$ is Cohen-Macaulay (see, for instance, Decker and de Jong (1998)). As a consequence, if $p_1, \ldots, p_n$ are primary invariants, then $K[\boldsymbol{x}]^G$ is a free $K[p_1, \ldots, p_n]$-module (see Bruns and Herzog (1993), Lemma 6.4.13). This gives information on the Hilbert series of $K[\boldsymbol{x}]^G$ (depending on the degrees of the primary invariants). Comparing with Molien's series, we get a polynomial from which the degrees of the secondary invariants and their number in each degree can be read off:

Remark 8.4. In the nonmodular case, the following hold:

(1) The degrees d_j of the secondary invariants and their number μ_j in each degree are uniquely determined as the exponents and the coefficients of the polynomial

$$\sum_{j=0}^{b} \mu_j t^{d_j} := H_{K[x]^G}(t) \cdot \prod_{i=1}^{n} (1 - t^{\deg(p_i)}).$$

(2) The total number of secondary invariants is

$$\frac{1}{|G|} \prod_{i=1}^{n} \deg(p_i)\ .$$

In fact, property (2) characterizes rings of invariants of finite groups which are Cohen-Macaulay (see Kemper (1996)). In the modular case, it can, thus, be used to check whether an explicitly given ring of invariants is Cohen-Macaulay.

□

Following Kemper and Steel (1999), we formulate an algorithm for computing secondary invariants such that the irreducible ones are singled out. Here, a secondary invariant is **irreducible** if it is not a power product of secondary invariants of lower degree.

Algorithm 8.5 (Secondary Invariants, Nonmodular Case).

INPUT:	the Reynolds operator $\mathcal{R}$, the Hilbert series $H = H_{K[x]^G}$, and a set $P = \{p_1, \ldots, p_n\}$ of primary invariants for $K[\boldsymbol{x}]^G$.
OUTPUT:	a corresponding system of secondary invariants, where the irreducible ones are singled out.

Step 1. Initialization:

- SecInv $:= \emptyset$;
- PowerProd $:= \{1\}$;
- compute the polynomial $\sum_{j=0}^{b} \mu_j t^{d_j} = H \cdot \prod_{i=1}^{n} (1 - t^{\deg(p_i)})$;

Step 2. Iteration (by degree):

```
for $j = 1, \ldots, b$ do
    $i := 0$;
    while $i < \mu_j$ do
      • for all power products $q \in K[\boldsymbol{x}]_{d_j}$ of elements of SecInv do
            if the classes $\overline{s}_1, \ldots, \overline{s}_i, \overline{q} \in K[\boldsymbol{x}]/\langle P\rangle$ are $K$-linearly inde-
            pendent then
                – $i := i + 1$;
                – $s_i := q$;
                – PowerProd := PowerProd $\cup \{s_i\}$;
      • compute the standard monomials $m_1, \ldots, m_c$
        of degree $d_j$ for $\langle P\rangle$;
      • for $r = 1, \ldots, c$ do
            if the classes $\overline{s}_1, \ldots, \overline{s}_i, \overline{\mathcal{R}(m_r)} \in K[\boldsymbol{x}]/\langle P\rangle$ are $K$-linearly
            independent then
                – $i := i + 1$;
                – $s_i := \mathcal{R}(m_r)$;
                – SecInv := SecInv $\cup \{s_i\}$;

return SecInv, PowerProd.
```
□

Note that if the set P of primary invariants has been found by Algorithm 8.3, then a Gröbner basis for the ideal $\langle P\rangle$ has already been computed for the dimension check. This Gröbner basis can be used to find the standard monomials in Step 2 above.

Example 8.6. In `SINGULAR`, rings of invariants of finite groups can be computed using the library `finvar.lib` written by Heydtmann (1999) and its command `invariant_ring`. As an example, we compute the ring of invariants of the Heisenberg group H_5 in its Schrödinger representation. That is, H_5 is the subgroup of $\mathrm{GL}_5(\mathbb{C})$ generated by the matrices

$$\sigma = \begin{pmatrix} 0&0&0&0&1 \\ 1&0&0&0&0 \\ 0&1&0&0&0 \\ 0&0&1&0&0 \\ 0&0&0&1&0 \end{pmatrix} \quad \text{and} \quad \tau = \begin{pmatrix} 1&0&0&0&0 \\ 0&\xi&0&0&0 \\ 0&0&\xi^2&0&0 \\ 0&0&0&\xi^3&0 \\ 0&0&0&0&\xi^4 \end{pmatrix}.$$

Here, $\xi = e^{\frac{2\pi i}{5}}$ is a primitive 5th root of unity in $\mathbb{C}$.

```
> LIB "finvar.lib";
> ring R = (0,a), (x(0..4)), dp;
> minpoly = a4+a3+a2+a+1;  // need fifth roots of unity
> matrix Si[5][5] = 0,0,0,0,1,
.                   1,0,0,0,0,
.                   0,1,0,0,0,
.                   0,0,1,0,0,
.                   0,0,0,1,0;
> matrix Ta[5][5];
> Ta[1,1] = 1;  Ta[2,2] = a;  Ta[3,3] = a2;  Ta[4,4] = a3;
> Ta[5,5] = a4;
> int aa = timer;          // time in seconds
> matrix P,S,IS = invariant_ring(Si,Ta,intvec(0,0,1));

Generating the entire matrix group. Whenever a new group element is
found, the corresponding ring homomorphism of the Reynolds operator
and the corresponding term of the Molien series is generated.

Group element 3 has been found.
                     [...]
Group element 125 has been found.

Now we are done calculating Molien series and Reynolds operator.

We can start looking for primary invariants...

Computing primary invariants in degree 5:
We find: x(0)*x(1)*x(2)*x(3)*x(4)
We find: x(0)^3*x(2)*x(3)+x(0)*x(1)*x(3)^3+x(0)*x(2)^3*x(4)+[...]
We find: x(0)*x(1)^3*x(2)+x(1)*x(2)^3*x(3)+x(0)^3*x(1)*x(4)+[...]
Computing primary invariants in degree 10:
We find: x(0)^10+x(1)^10+x(2)^10+x(3)^10+x(4)^10
We find: -x(0)^5*x(1)^5+x(0)^5*x(2)^5-[...]

We found all primary invariants.

Polynomial telling us where to look for secondary invariants:
 x(0)^30+3*x(0)^25+24*x(0)^20+44*x(0)^15+24*x(0)^10+3*x(0)^5+1

In degree 0 we have: 1

Searching in degree 5, we need to find 3 invariant(s)...
We find: x(0)^2*x(1)^2*x(3)+x(0)*x(2)^2*x(3)^2+[...]
We find: x(0)^2*x(1)*x(2)^2+x(1)^2*x(2)*x(3)^2+[...]
We find: x(0)^5+x(1)^5+x(2)^5+x(3)^5+x(4)^5 .

Searching in degree 10, we need to find 24 invariant(s)...
                     [...]
Searching in degree 15, we need to find 44 invariant(s)...
```

```
                      [...]
Searching in degree 20, we need to find 24 invariant(s)...
                      [...]
Searching in degree 25, we need to find 3 invariant(s)...
                      [...]
Searching in degree 30, we need to find 1 invariant(s)...
We find: x(0)^15*x(1)^10*x(3)^5+x(0)^10*x(1)^15*x(3)^5+[...]

We're done!

> timer-aa;
1554
```

To just compute a vector space basis for the invariants of a given degree with `finvar.lib`, use the command `invariant_basis`. The famous Horrocks-Mumford quintics, for instance, form a $\mathbb{C}$-basis for the degree-5 invariants of H_5 (see Horrocks and Mumford (1973)):

```
> aa = timer;
> ideal HMQ = invariant_basis(5,Si,Ta);
> timer-aa;
2
> print(HMQ);
x(0)*x(1)*x(2)*x(3)*x(4),
x(0)^2*x(1)*x(2)^2+x(1)^2*x(2)*x(3)^2+x(0)^2*x(3)^2*x(4)+[...]
x(0)^2*x(1)^2*x(3)+x(0)*x(2)^2*x(3)^2+x(1)^2*x(2)^2*x(4)+[...]
x(0)^3*x(2)*x(3)+x(0)*x(1)*x(3)^3+x(0)*x(2)^3*x(4)+[...]
x(0)*x(1)^3*x(2)+x(1)*x(2)^3*x(3)+x(0)^3*x(1)*x(4)+[...]
x(0)^5+x(1)^5+x(2)^5+x(3)^5+x(4)^5
```
□

Remark 8.7. Following Remark 4.12 in Lecture 4, we may ease the computations in the example above by working over a finite field K. In the session below, we pick the field $K = \mathbb{F}_{101}$ for which 36 is a primitive 5th root of unity.

```
> LIB "finvar.lib";
> ring R = 101, (x(0..4)), dp;
> matrix Si[5][5] = 0,0,0,0,1,
.                   1,0,0,0,0,
.                   0,1,0,0,0,
.                   0,0,1,0,0,
.                   0,0,0,1,0;
> number a = 36;              // primitive fifth root of unity
> matrix Ta[5][5];
> Ta[1,1] = 1;  Ta[2,2] = a;  Ta[3,3] = a^2;  Ta[4,4] = a^3;
> Ta[5,5] = a^4;
> int aa = timer;             // time in seconds
> matrix P,S,IS = invariant_ring(Si,Ta,intvec(0,0,0));
> timer-aa;
456
```

```
> size(S);
100
> print(S[100]);
[x(0)^15*x(1)^10*x(3)^5+x(0)^10*x(1)^15*x(3)^5+[...]
> ideal HMQ = invariant_basis(5,Si,Ta);
```
□

At this point, we set the computation of invariants aside and consider for a moment a geometric application of what we did above. Addressing the readers who are familiar with the classification of vector bundles on projective spaces, we remark that the geometric significance of the Heisenberg group H_5 comes from the fact that its normalizer in $\mathrm{SL}_5(\mathbb{C})$ is the symmetry group of the Horrocks-Mumford bundle on $\mathbb{P}^4 = \mathbb{P}^4(\mathbb{C})$ (see Horrocks and Mumford (1973) and Decker (1984)). In the following more involved example, which is in the spirit of Exercise 1.5 and Examples 4.13 and 4.14 in Lecture 4, we present an experiment which leads to the construction of the Horrocks-Mumford bundle and which reveals some of the geometry behind it:

Example 8.8. Continuing our SINGULAR session from the remark above, we compute a primary decomposition of the ideal generated by the first two Horrocks-Mumford quintics.

```
> ring P4 = 101, (x(0..4)), dp;
> ideal HMQ = fetch(R,HMQ);
> ideal CI = HMQ[1], HMQ[2];
> list DEG = primdecGTZ(CI);
> size(DEG);
10
```

We see that that there are ten components. In fact, there are five components defining a cubic surface and five components defining a double structure on a plane. For instance:

```
> ideal I6 = DEG[6][1]; I6;
I6[1]=x(1)^2*x(3)+x(2)*x(4)^2
I6[2]=x(0)
> degree(std(I6));
// dimension (proj.)  = 2
// degree (proj.)   = 3
> ideal I10 = DEG[10][1]; I10;
I10[1]=x(2)^2
I10[2]=x(2)*x(3)^2+x(0)*x(4)^2
I10[3]=x(0)*x(2)
I10[4]=x(0)^2
> degree(std(I10));
// dimension (proj.)  = 2
// degree (proj.)   = 2
```

The union of the degree 2 surfaces defines a degree 10 subscheme X of $\mathbb{P}^4$. We compute the minimal free resolution of its coordinate ring:

```
> ideal IX = intersect(DEG[3][1],DEG[4][1],DEG[8][1],DEG[9][1],
.                     DEG[10][1]);
> degree(std(IX));
// dimension (proj.)  = 2
// degree (proj.)   = 10
> resolution FIX = mres(IX,0);
> print(betti(FIX),"betti");
           0     1     2     3     4
------------------------------------
    0:     1     -     -     -     -
    1:     -     -     -     -     -
    2:     -     -     -     -     -
    3:     -     -     -     -     -
    4:     -     3     -     -     -
    5:     -    15    35    20     -
    6:     -     -     -     -     2
------------------------------------
total:     1    18    35    20     2
```

The degree 10 subscheme X is one of the degenerated Horrocks-Mumford surfaces studied by Barth et al (1987). It, thus, arises as a zero scheme of the Horrocks-Mumford bundle. In other words, there is an exact sequence

$$0 \longleftarrow \mathrm{I}(X)(5) \longleftarrow M_{\mathrm{HM}} \longleftarrow S = \mathbb{C}[x_0, \ldots, x_4] \longleftarrow 0\,, \qquad (8.1)$$

where $\mathrm{I}(X)(5)$ is the 5th twist of $\mathrm{I}(X)$, and where M_{HM} is the graded module of global sections of the Horrocks-Mumford bundle. We expect from (8.1) and the Betti diagram visualizing the minimal free resolution of the coordinate ring of $\mathrm{I}(X)$ that M_{HM} has a minimal free resolution of type

$$0 \longleftarrow M_{\mathrm{HM}} \longleftarrow 4S \oplus 15S(-1) \longleftarrow 35S(-2) \longleftarrow 20S(-3) \longleftarrow 2S(-5) \longleftarrow 0\,.$$

It turns out that we may compute this resolution as follows:

```
> module N = transpose(FIX[3]);
> homog(N);
1
> intvec deg_N = attrib(N,"isHomog");
> attrib(N,"isHomog",deg_N-3);        // set degrees
> resolution FN = mres(N,0);
> print(betti(FN),"betti");
           0     1     2     3     4     5
------------------------------------------
   -3:    20    35    15     -     -     -
   -2:     -     -     4     -     -     -
   -1:     -     -     -     -     -     -
    0:     -     -     5    15    10     2
------------------------------------------
total:    20    35    24    15    10     2
```

```
> matrix NN = FN[2];
> matrix PRESMHM[35][19] = NN[1..35,1..19];
> PRESMHM = transpose(PRESMHM);
> resolution FMHM = mres(PRESMHM,0);
> print(betti(FMHM),"betti");
           0     1     2     3
------------------------------
    0:     4     -     -     -
    1:    15    35    20     -
    2:     -     -     -     2
------------------------------
total:    19    35    20     2
```

Having constructed the Horrocks-Mumford bundle starting from a zero scheme of one of its sections, we may now construct zero schemes of other sections. The ideal sheaf of each such scheme fits into an exact sequence of type (8.1) and can, thus, be computed by a cokernel construction (see Lecture 4). As shown by Horrocks and Mumford (1973) with a nontrivial proof, a generic section of the Horrocks-Mumford bundle vanishes along a smooth surface (of degree 10). We verify this using `SINGULAR` (applying the Jacobian criterion as in Examples 4.13 and 4.14 in Lecture 4):

```
> matrix zero[1][15];
> matrix ran = random(100,1,4);
> matrix psi = transpose(concat(zero,ran));
> matrix pres = PRESMHM + module(psi);
> module dir = transpose(pres);
> resolution fdir = mres(dir,2);
> print(betti(fdir),"betti");
           0     1     2
------------------------
    0:    35    15     -
    1:     -     3     -
    2:     -     -     -
    3:     -     -     -
    4:     -     -     -
    5:     -     -     1
------------------------
total:    35    18     1
> ideal IA = groebner(flatten(fdir[2]));
> int codimIA = nvars(P4) - dim(IA);
> ideal sIA = minor(jacob(IA),codimIA)+IA;
> nvars(P4) - dim(groebner(sIA));
5
```

We randomly choose two quintics in the ideal defining A. The complete intersection surface defined by these quintics is the union of A and some other surface, say A' (we say that A is **linked** to A' by the complete intersection). We compute A' via saturation and check that it is smooth:

```
> matrix dummy[1][3] = IA[1..3];     // the 3 quintics in IA
> ideal CI2 = dummy*random(100,3,2);
> ideal IA' = sat(CI2,IA)[1];
> resolution FIA' = mres(IA',0);
> print(betti(FIA'),"betti");
           0     1     2     3     4
------------------------------------
    0:     1     -     -     -     -
    1:     -     -     -     -     -
    2:     -     -     -     -     -
    3:     -     -     -     -     -
    4:     -     3     -     -     -
    5:     -     -     -     -     -
    6:     -     5    15    10     2
------------------------------------
total:     1     8    15    10     2
> ideal sIA' = slocus(IA');
> nvars(P4) - dim(std(sIA'));
5
```

It follows from general classification results that A is an Abelian surface and that A' is the blow-up of an Abelian surface in 25 points (see Aure et al (1997), Section 2 and the references cited there). In fact, there are 25 exceptional lines on A' which can be rediscovered as the 6-secants to A. By Bézout's theorem, the 6-secants to A must be contained in the hypersurfaces defined by the quintics in the ideal of A. In fact, these quintics cut out A and the 25 lines:

```
> ideal QA = IA[1..3];
> ideal HMlines = sat(QA,IA)[1];     // result is a Groebner basis
> degree(HMlines);
// dimension (proj.)  = 1
// degree (proj.)   = 25
```
□

We refer to Popescu (1993) and Aure et al (1997) for further surfaces admitting a H_5-symmetry.

8.1.2 The Modular Case

Still suppose that G is a finite group, explicitly given as a matrix group $G \subset \mathrm{GL}_n(K)$ by a (finite) set of generating matrices. Linear algebra provides an alternative way of computing a K-basis for $K[\boldsymbol{x}]_d^G$, where d is a given degree. Indeed, to be invariant under just one element of G imposes a linear condition on the polynomials of degree d. Thus, we may obtain the desired K-basis by solving a linear system of equations depending on the given generators for G; no Reynolds operator and no precomputed Hilbert series are needed. In particular, we can use Algorithm 8.3 to compute primary invariants also in the modular case. The computation of secondary invariants in the modular case is reduced to that in the nonmodular case by a trick of Kemper (1996).

In describing the resulting algorithm, we suppose that a system $p_1, \dots, p_n$ of primary invariants for G has already been computed (for instance, by applying Algorithm 8.3).

Step 1. Choose a subgroup H of G whose order is invertible in K (for example, the trivial subgroup will do).

Then $p_1, \dots, p_n$ is also a system of primary invariants for H.

Step 2. Compute $t_1, \dots, t_k$, a system of secondary invariants for H corresponding to $p_1, \dots, p_n$ (for instance, by applying Algorithm 8.5).

Step 3. Choose a subset $\Delta \subset G$ such that the elements of $\Delta \cup H$ generate G.

Note that $K[\boldsymbol{x}]$ is a free $K[p_1, \dots, p_n]$-module of rank $\prod_{i=1}^n \deg(p_i)$ with a basis $\mathcal{B}$ given by the standard monomials for $\langle p_1, \dots, p_n\rangle$ (with respect to the chosen monomial order). Setting $r = |\Delta| \cdot \prod_i \deg(p_i)$, we get a commutative diagram of graded $K[p_1, \dots, p_n]$-modules with exact rows and columns

$$\begin{tikzcd}
 & 0 & 0 & 0 \\
0 \arrow[r] & K[\boldsymbol{x}]^G \arrow[r] \arrow[u, dashed] & K[\boldsymbol{x}]^H \arrow[r, "\alpha"] \arrow[u] & \bigoplus_{\pi\in\Delta} K[\boldsymbol{x}] \arrow[u] \\
0 \arrow[r, dashed] & M \arrow[r, dashed] \arrow[u, dashed] & K[p_1,\dots,p_n]^k \arrow[r, dashed, "\widetilde{\alpha}"'] \arrow[u, "\beta"'] & K[p_1,\dots,p_n]^r \arrow[u] \\
 & 0 \arrow[u, dashed] & 0 \arrow[u] & 0\,. \arrow[u]
\end{tikzcd}$$

Here, α is the map defined by $f \mapsto (\pi f - f)_{\pi\in\Delta}$, and β is the map given by $(f_1, \dots, f_k) \mapsto \sum_{j=1}^k f_j t_j$.

Step 4. Via linear algebra or Gröbner bases, compute the map $\widetilde{\alpha}$, that is, the representation of each $(\pi t_j - t_j)$ in terms of $\mathcal{B}$.

Step 5. Compute the first syzygy module M of $\operatorname{im}\widetilde{\alpha} \subset K[p_1, \dots, p_n]^r$, and get the secondary invariants for G by determining the images in $K[\boldsymbol{x}]^H$ of the generators for M.

Example 8.9. We consider the cyclic group G of order 4, represented as a subgroup of $\mathrm{GL}_4(\mathbb{F}_2)$.

```
> 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;
> print(A);
0,0,0,1,
1,0,0,0,
0,1,0,0,
0,0,1,0
```

```
> LIB "finvar.lib";
> matrix P,S = invariant_ring(A);
> P;
P[1,1]=x(1)+x(2)+x(3)+x(4)
P[1,2]=x(1)*x(3)+x(2)*x(4)
P[1,3]=x(1)*x(2)+x(2)*x(3)+x(1)*x(4)+x(3)*x(4)
P[1,4]=x(1)*x(2)*x(3)*x(4)
> size(S);
5
```

We, hence, need more than $\prod_{i=1}^{4}\deg(p_i)/|G| = 4$ secondary invariants. So, $K[\boldsymbol{x}]^G$ is not Cohen-Macaulay. A check on the secondary invariants shows that the invariants of degree $\leq |G| = 4$ do not generate $K[\boldsymbol{x}]^G$ (see Exercise 5.1). Thus, Noether's degree bound (1916)[1] does not hold in the modular case (see Richman (1990) for examples with a fixed group order involving generators of arbitrary high degree). □

8.1.3 Quotients for Finite Group Actions

We begin by pointing out that the algorithms presented so far in this section do not compute a **fundamental system of invariants**. That is, they do not compute a minimal generating set for the ring of invariants under consideration. As a solution to Exercise 5.1, we will write a procedure `min_generating_set`

- which takes as input matrices of primary and secondary invariants of a finite group as provided by the `invariant_ring` command, and
- which computes a **fundamental system of invariants**.

The following example shows this procedure at work. Further, we show how to compute the algebra relations on the resulting invariants.

Example 8.10. We consider the cyclic group $\mathbb{Z}_5$ of order 5, represented as the subgroup of $\mathrm{GL}_4(\mathbb{C})$ with generator

$$A = \begin{pmatrix} \xi & 0 & 0 & 0 \\ 0 & \xi^2 & 0 & 0 \\ 0 & 0 & \xi^3 & 0 \\ 0 & 0 & 0 & \xi^4 \end{pmatrix}.$$

Here, as in Example 8.6, $\xi = e^{\frac{2\pi i}{5}}$. We compute the ring of invariants using `SINGULAR`:

```
> LIB "finvar.lib";
> ring R = (0,a), (x(0..3)), dp;
> minpoly = a4+a3+a2+a+1;
```

[1] In the nonmodular case, Noether proved that $K[\boldsymbol{x}]^G$ is generated in degrees $\leq |G|$.

```
> matrix A[4][4];
> A[1,1] = a;  A[2,2] = a2;  A[3,3] = a3;  A[4,4] = a4;
> matrix P,S,IS = invariant_ring(A,intvec(0,0,0));
> size(P);
4
> size(S);
12
```

Now we enter the procedure `min_generating_set` (without printing this here) and apply it to `P` and `S`. As a result, we obtain a fundamental system of invariants for $\mathbb{C}[x_0,\ldots,x_3]^{\mathbb{Z}_5}$:

```
> ideal FSI = min_generating_set(P,S);
> size(FSI);
14
```

Next, we compute the $\mathbb{C}$-algebra relations on the minimal generators as elements in a polynomial ring over $\mathbb{C}$ with 14 variables, say $y_0,\ldots,y_{13}$. In this way, we represent $\mathbb{C}[x_0,\ldots,x_3]^{\mathbb{Z}_5}$ as the quotient of $\mathbb{C}[y_0,\ldots,y_{13}]$ modulo the ideal defined by the relations. Observe, however, that this ideal is not homogeneous with respect to the natural grading of $\mathbb{C}[y_0,\ldots,y_{13}]$. Indeed, the fundamental generators have different degrees. To represent $\mathbb{C}[x_0,\ldots,x_3]^{\mathbb{Z}_5}$ as a graded ring, we have to assign the degrees of the fundamental generators as weights to the variables y_i:

```
> ring Rnew = 0, (x(0..3)), dp;  // coefficient field is now Q
> ideal FSI = fetch(R,FSI);
> ideal ZERO;
> ring R1 = 0, (y(0..13)), wp(5,5,5,5,4,4,4,4,3,3,3,3,2,2);
> ideal REL = preimage(Rnew,FSI,ZERO);
> homog(REL);                    // check that REL is homogeneous
1
> size(REL);
54
```

Inspecting the fundamental generators, we see that the Fermat quintic

$$F = x_0^5 + x_1^5 + x_2^5 + x_3^5$$

is one of the invariants:

```
> setring Rnew;
> FSI[4];
x(0)^5+x(1)^5+x(2)^5+x(3)^5
```

Thus, the action of $\mathbb{Z}_5$ on $\mathbb{C}[x_0,\ldots,x_3]$ induces an action of $\mathbb{Z}_5$ on the affine ring $S = \mathbb{C}[x_0,\ldots,x_3]/\langle F\rangle$ and it makes sense to speak of the ring $S^{\mathbb{Z}_5}$ of invariants under this action.

Note that the natural projection $\mathbb{C}[x_0,\ldots,x_3]^{\mathbb{Z}_5} \to S^{\mathbb{Z}_5}$ is surjective. Indeed, if $f \in \mathbb{C}[x_0,\ldots,x_3]$ is a polynomial representing an invariant of S under $\mathbb{Z}_5$ and $\mathcal{R} : \mathbb{C}[x_0,\ldots,x_3] \to \mathbb{C}[x_0,\ldots,x_3]^{\mathbb{Z}_5}$ is the Reynolds operator, then $\mathcal{R}(f)$ is congruent to f mod F. We represent $S^{\mathbb{Z}_5}$ as a quotient of the weighted polynomial ring $\mathbb{C}[y_0,\ldots,y_{13}]$:

```
> ideal F = FSI[4];
> setring R1;
> ideal GODEAUX = preimage(Rnew,FSI,F);
> size(GODEAUX);
55
> GODEAUX[1];
y(3)
> dim(std(GODEAUX));
3
```
$\square$

Readers who are familiar with the classification of surfaces may have realized that in the example above, we used SINGULAR to repeat a classical construction due to Godeaux (see Barth et al (2004), Sections V.15 and VII.10 and the references cited there). Namely, from a geometric point of view, we constructed the quotient of the quintic hypersurface $V(x_0^5 + x_1^5 + x_2^5 + x_3^5) \subset \mathbb{P}^3$ under the action of $\mathbb{Z}_5$ described above. This quotient is a smooth surface of general type which is usually referred to as a **Godeaux surface** (the smoothness follows from the fact that $\mathbb{Z}_5$ acts freely on $V(x_0^5+x_1^5+x_2^5+x_3^5)$, see Mumford (1970), Section II.7).

The Godeaux surface is one interesting example of an algebraic set A whose construction yields an embedding of A into a **weighted projective space** (in our case, this is the space $\mathbb{P}(5,5,5,5,4,4,4,4,3,3,3,3,2,2)$). We may, then, study A in terms of this ambient space, for instance, by projecting it to weighted projective subspaces. See Harris (1992), Chapter 10 and Dolgachev (1982) for weighted projective spaces.

The construction of "geometric" quotients for group actions on algebraic sets is the subject of geometric invariant theory (see Mumford et al (1994) and Newstead (1978)). In general, such quotients do not exist. In the case of finite groups acting on affine or projective algebraic sets, however, the situation is different (see Harris (1992), Chapter 10 and Mumford (1970), Section II.7). Note that weighted projective spaces arise as quotients of specific finite group actions on classical projective spaces.

8.2 Linearly Reductive Groups

In Lecture 1, we introduced some of the basic concepts and theorems of computational algebraic geometry by historical remarks, starting with papers originating from classical invariant theory. In particular, we quickly reviewed Hilbert's landmark papers (1890, 1893) in which classical invariant theory culminated and in which Hilbert showed that a large class of rings of invariants is finitely generated. In modern terminology, he showed that if G is a linearly reductive group acting rationally on the K-vector space of linear forms in $K[\boldsymbol{x}]$, then $K[\boldsymbol{x}]^G$ is finitely generated.

We will not recall what a linearly reductive group and a rational action of such a group are (see, for instance, Derksen and Kemper (2002) for precise definitions). We should mention, however, that if such an action on the space of linear forms in $K[\boldsymbol{x}]$ is given, then, as for finite groups in the nonmodular case, there exists a **Reynolds operator** $\mathcal{R} : K[\boldsymbol{x}] \to K[\boldsymbol{x}]$ projecting $K[\boldsymbol{x}]$ onto $K[\boldsymbol{x}]^G$. Also, the ring $K[\boldsymbol{x}]^G$ is Cohen-Macaulay by a result of Hochster and Roberts (1974). Examples of linearly reductive groups are finite groups in the nonmodular case and the classical groups $\mathrm{SL}_n(\mathbb{C})$, $\mathrm{GL}_n(\mathbb{C})$, $\mathrm{O}_n(\mathbb{C})$, $\mathrm{SO}_n(\mathbb{C})$ und $\mathrm{Sp}_n(\mathbb{C})$.

Hilbert's first proof of the finiteness result (1890) is an application of the basis theorem which was proved for that purpose. In fact, we may conclude from the basis theorem that the ideal $I_{\mathcal{N}}$ generated by all homogeneous invariants of positive degree is generated by finitely many of them, say by $f_1, \dots, f_r$. That is, if f is any homogeneous invariant of positive degree, then f can be written as a $K[\boldsymbol{x}]$-linear combination $f = \sum_{i=1}^r g_i f_i$. Applying the Reynolds operator $\mathcal{R}$, we get

$$f = \sum_{i=1}^{r} \mathcal{R}(g_i) f_i.$$

The $\mathcal{R}(g_i)$ are invariants, which may be assumed to be of lower degree than f. By induction on the degree, the $\mathcal{R}(g_i)$ are in the subalgebra generated by $f_1, \dots, f_r$. Hence, so is f.

As it stands, this proof is nonconstructive since no recipe is given for computing the ideal generators $f_1, \dots, f_r$. Only recently, Derksen (1999) refined Hilbert's ideas and turned them into an algorithm which reduces the problem to elimination, and which is easy to implement.

Hilbert himself, in his more constructive second paper (1893), presented two ways of reducing the computation of invariants to normalization (see Sturmfels (1993) and Decker and de Jong (1998)). For this purpose, he proved the Hilbert-Mumford criterion (which, in practical terms, requires elimination), the Nullstellensatz, and the Noether normalization theorem (in the graded case). Except for rather small examples, Derksen's and Hilbert's algorithm are not of practical interest (requiring a Gröbner basis computation with respect to an elimination order in a large number of variables).

Remark 8.11 (Further Reading). For more details and proofs of the results presented in this lecture, see Decker and de Jong (1998), Derksen and Kemper (2002), and Sturmfels (1993).

Lecture 9

Computing in Local Rings

This lecture deals with methods for studying the behavior of a given algebraic set A "near" a given point $p \in A$, that is, for studying arbitrarily small neighborhoods of p in A. To give this a precise meaning, we introduce the following notation.

Definition 9.1. Let X be a topological space, and let $p \in X$ be a point. We call two subsets $A, B \subset X$ equivalent at p if there is a neighborhood U of p in X such that $A \cap U = B \cap U$ and refer to the equivalence class of A as the **germ of A at p**, written (A, p). $\square$

In the geometric setting considered in this lecture, A will always be an algebraic subset of $\mathbb{A}^n = \mathbb{A}^n(\overline{K})$, where K is a field and where $\overline{K}$ is an algebraically closed field containing K. We suppose that A is defined by polynomials in $K[\boldsymbol{x}] = K[x_1, \dots, x_n]$ and that $p = (0, \dots, 0) \in \mathbb{A}^n$ is the origin. Choosing different topologies on $\mathbb{A}^n$, we get different types of germs (A, p) of A at p. Depending on the choice of topology, we associate to (A, p) a local ring of germs of functions at p, where germs of functions at p are defined as above by identifying two functions defined on A near p if their restrictions to a suitable neighborhood of p in A coincide. We may say, then, that studying A near p means to study properties of the germ of A at p and its associated local ring.

If we choose the Zariski topology on $\mathbb{A}^n$, we refer to the resulting germ (A, p) as the **algebraic germ** of A at p and associate to it the **ring of germs of regular functions** at p obtained from $K[A]$ by inverting all polynomial functions on A which do not vanish at p. Formally, this ring is the localization of $\overline{K}[A]$ at the maximal ideal $\langle \overline{\boldsymbol{x}} \rangle$ of polynomial functions on A vanishing at p,

$$\mathcal{O}_{A,p} = \overline{K}[A]_{\langle \overline{\boldsymbol{x}} \rangle} = \overline{K}[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle} / \mathrm{I}(A)\, \overline{K}[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle}$$

(see Remark 2.12 in Lecture 2). Note that two algebraic sets $A, B \subset \mathbb{A}^n$ define the same germ at p with respect to the Zariski topology iff their irreducible components through p coincide. Algebraically, if $I \subset K[\boldsymbol{x}]$ is any ideal defining

A, and if Q is a primary component of I such that $Q \not\subset \langle \boldsymbol{x} \rangle$, then Q contains a unit of $K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle}$. Hence, $QK[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle} = K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle}$.

If $\overline{K} = \mathbb{C}$, we may also choose the usual Euclidean topology on $\mathbb{A}^n$. In this case, we refer to the resulting germ (A, p) as the **analytic germ** of A at p and associate to it the **ring of germs of holomorphic functions** at p,

$$\mathcal{O}_{A,p}^{\mathrm{an}} := \mathbb{C}\{\boldsymbol{x}\}/\mathrm{I}(A)\,\mathbb{C}\{\boldsymbol{x}\}\,.$$

Here, $\mathbb{C}\{\boldsymbol{x}\}$ is the ring of convergent power series in $x_1, \dots, x_n$ with coefficients in $\mathbb{C}$.

Over an arbitrary field K, there is no analogue to the Euclidean topology, and it is not meaningful to speak of convergent power series. We may, however, always consider the ring

$$\widehat{\mathcal{O}}_{A,p} := \overline{K}[[\boldsymbol{x}]]/\mathrm{I}(A)\,\overline{K}[[\boldsymbol{x}]]\,,$$

where $\overline{K}[[\boldsymbol{x}]]$ is the ring of formal power series in $x_1, \dots, x_n$ with coefficients in $\overline{K}$. Its geometric counterpart is defined in the language of schemes and referred to as the **formal germ** of A at p (we go no further into this).

Note that the neighborhoods of p in $\mathbb{A}^n$ in the Zariski topology are rather large (in fact, dense in $\mathbb{A}^n$), whereas in the Euclidean topology, we can choose arbitrarily small ε-neighborhoods of p in $\mathbb{A}^n$. On the algebraic side, this is reflected by the fact that there are more holomorphic functions at p than regular functions at p. In fact, there are strict inclusions of rings

$$\mathbb{C}[A] \subsetneq \mathcal{O}_{A,p} \subsetneq \mathcal{O}_{A,p}^{\mathrm{an}} \subsetneq \widehat{\mathcal{O}}_{A,p}\,,$$

where for the first inclusion to hold we have to suppose that all components of A pass through p.

Remark 9.2. (1) We always have the natural inclusion $K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle} \subset K[[\boldsymbol{x}]]$ obtained by power series expansion. Indeed, write each element of $K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle}$ as a fraction of type $g/(1-h)$, with polynomials $g \in K[\boldsymbol{x}]$ and $h \in \langle \boldsymbol{x} \rangle$, and expand $1/(1-h)$ in a geometric series:

$$\frac{g}{1-h} = \sum_{k=0}^{\infty} gh^k\,.$$

This defines a formal power series as the sequence $(gh^k)_{k \in \mathbb{N}}$ converges to zero with respect to the **$\langle \boldsymbol{x} \rangle$-adic topology**. That is, for each $N \in \mathbb{N}$, there is some $k_0 \in \mathbb{N}$ such that $gh^k \in \langle \boldsymbol{x} \rangle^N$ for all $k \geq k_0$. If $K \subset \mathbb{C}$, we obtain a convergent power series in this way.

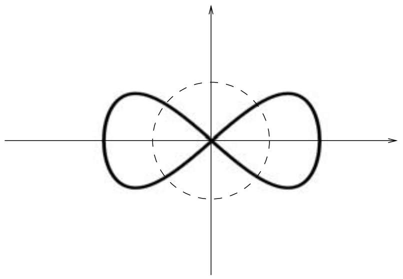
(2) If $I \subset K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle}$ is an ideal, then the induced homomorphism $K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle}/I \to K[[\boldsymbol{x}]]/IK[[\boldsymbol{x}]]$ is injective. Indeed, the inclusion of rings $K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle} \subset K[[\boldsymbol{x}]]$ is faithfully flat (see Proposition 9.4 (1) below for a definition and for a more general statement). In particular, $IK[[\boldsymbol{x}]] \cap K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle} = I$ (Matsumura (1986), Theorem 7.5). □

Investigating A near p means, in particular, to decide whether A is smooth at p and, if not, to study the singularity of A at p. The computational questions arising in this context include:

- Compute invariants of the germ (A, p) that help to describe the type of singularity of A at p. Most notably, compute the Tjurina number and the Milnor number.
- Decide whether (A, p) is irreducible, that is, whether $\langle 0 \rangle$ is a prime ideal of the associated local ring.
- If (A, p) is reducible, compute the number of irreducible components, that is, the number of (minimal) associated primes of $\langle 0 \rangle$ in the associated local ring.
- Check whether the local ring associated to (A, p) is Cohen-Macaulay or whether a given map of germs is flat.
- Study the behavior of (A, p) under a small flat deformation.

The answer to many local questions does not depend on the type of germ chosen (for the Cohen-Macaulay property, for instance, this holds due to Lecture 5, Proposition 5.37). If it comes to local irreducibility, however, the situation is different:

Example 9.3. Consider the complex affine plane curve $C = \mathrm{V}\big(x^2(1-x^2) - y^2\big)$ at the origin:



The picture indicates:

- In the affine plane and, thus, in each Zariski neighborhood of the origin, C is irreducible.
- In a sufficiently small Euclidean neighborhood of the origin, however, C has two smooth branches that meet transversally.

Algebraically, the polynomial $f = x^2(1-x^2) - y^2$ is irreducible in $\mathbb{C}[x, y]$ (for instance, use `absFactorize` to check this). In contrast, considering f as an element of $\mathbb{C}\{x, y\}$, we have the nontrivial decomposition

$$f = \Big(x\sqrt{1-x^2} - y\Big)\Big(x\sqrt{1-x^2} + y\Big). \qquad \square$$

To provide tools for local computations, we introduce **standard bases**, extending the concept of Gröbner bases to the local rings $K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle}$, $K\{\boldsymbol{x}\}$ (in case $K \subset \mathbb{C}$), and $K[[\boldsymbol{x}]]$. Note that from a practical point of view, we are essentially restricted to computations over $K[\boldsymbol{x}]$ and $K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle}$ (or quotient rings thereof). This is often sufficient, however, to study analytic and formal germs, too. The following proposition, for instance, is the key to computing invariants of such germs:

Proposition 9.4. *Let $I \subset K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle}$ be an ideal. Then:*

(1) The inclusion $K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle}/I \subset K[[\boldsymbol{x}]]/IK[[\boldsymbol{x}]]$ is **faithfully flat**. *This is to say, a sequence $0 \to M' \to M \to M'' \to 0$ of $K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle}/I$-modules is exact iff the sequence of $K[[\boldsymbol{x}]]/IK[[\boldsymbol{x}]]$-modules obtained by tensorizing with $K[[\boldsymbol{x}]]/IK[[\boldsymbol{x}]]$ is exact.*

(2) If $\dim_K(K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle}/I) < \infty$, the inclusion $K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle}/I \subset K[[\boldsymbol{x}]]/IK[[\boldsymbol{x}]]$ is an isomorphism of local K-algebras. In particular, both vector spaces have the same dimension.

If $K \subset \mathbb{C}$, the analogous statements hold for $K\{\boldsymbol{x}\}$ in place of $K[[\boldsymbol{x}]]$.

The key step in the proof of assertion (1) is to show that with the notation and under the assumptions of Lecture 5, Theorem 5.11, a finitely generated S-module M is flat over R iff $M/\mathfrak{m}^n M$ is flat over $R/\mathfrak{m}^n$ for all n (using Krull's intersection theorem). See Greuel and Pfister (2002), Corollaries 7.4.4, 7.4.6 and Exercise 7.3.12 for details. For (2), observe that due to the assumption, a sufficiently large power of the maximal ideal $\langle \boldsymbol{x} \rangle \subset K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle}$ is contained in I. Thus, each element of $K[[\boldsymbol{x}]]/IK[[\boldsymbol{x}]]$ may be represented by a polynomial.

We present the tools for local computations in six sections. In Section 9.1, we treat local and mixed monomial orders. Section 9.2 is concerned with the definition and computation of standard bases. In Section 9.3, we return to factorization and primary decomposition, now interpreting the results of our computations in terms of the local ring $K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle}$. Section 9.4 is concerned with computing local invariants such as the local dimension, the Milnor number, and the Tjurina number. Here, Proposition 9.4 above applies, that is, standard basis computations yield results for the algebraic germ and for the analytic germ as well. In Section 9.5, we treat the elimination problem in local rings. Finally, and somewhat independently, Section 9.6 on Hamburger-Noether expansions is devoted to the factorization problem in $K[[x, y]]$ (geometrically, to computing information on the branches of germs of plane curves).

9.1 Rings Implemented by Monomial Orders

Given any monomial order $>$ on $K[\boldsymbol{x}]$, the set

$$S_> := \{u \in K[\boldsymbol{x}] \mid \mathrm{L}_>(u) \in K \setminus \{0\}\}$$

is multiplicatively closed. We define $K[\boldsymbol{x}]_>$ to be the localization of $K[\boldsymbol{x}]$ at $S_>$,

$$K[\boldsymbol{x}]_> := S_>^{-1}K[\boldsymbol{x}] = \left\{ \frac{f}{u} \;\middle|\; f, u \in K[\boldsymbol{x}],\, \mathrm{L}_>(u) \in K \setminus \{0\} \right\} .$$

In `SINGULAR`, it is possible to compute with ideals of $K[\boldsymbol{x}]_>$ and submodules of free $K[\boldsymbol{x}]_>$-modules. In fact, computations over $K[\boldsymbol{x}]_>$ are mimicked by considering the elements of $S_>$ as units in standard basis computations which entirely take place in $K[\boldsymbol{x}]$. To emphasize this point, we say that $K[\boldsymbol{x}]_>$ is the ring **implemented by the monomial order $>$**. Before explaining in detail how different types of orders are related to different types of rings, we illustrate the behavior of standard basis computations by a simple example:

Example 9.5. We consider the ideal generated by $x^2 + x = x(x+1)$ in two different rings:

```
> ring R = 0, x, dp;    // global order: x>1
> ideal I = x2+x;
> std(I);
_[1]=x2+x
> ring S = 0, x, ds;    // local order: 1>x
> ideal I = x2+x;
> std(I);
_[1]=x
```
□

Clearly, if $>$ is global, then $S_> = K \setminus \{0\}$ and $K[\boldsymbol{x}]_> = K[\boldsymbol{x}]$ (in `SINGULAR`, the `p` in the names of the predefined global monomial orders such as `lp` and `dp` stands for "polynomial ring").

Local Monomial Orders

The name *local* monomial order comes from the fact that these orders are precisely the orders implementing the *local* ring $K[\boldsymbol{x}]_{\langle x\rangle}$. This is to say, if $>$ is a monomial order on $K[\boldsymbol{x}]$, the following statements are equivalent:

(1) $>$ is **local**, that is, $1 > x_i$ for $i = 1, \dots, n$.
(2) $K[\boldsymbol{x}]_> = K[\boldsymbol{x}]_{\langle \boldsymbol{x}\rangle}$.

Note that every local monomial order is obtained by "inverting" a global one. Indeed, if $>'$ is a monomial order on $K[\boldsymbol{x}]$, and if $>$ is defined by setting $\boldsymbol{x}^\alpha > \boldsymbol{x}^\beta$ iff $\boldsymbol{x}^\beta >' \boldsymbol{x}^\alpha$, then $>$ is local iff $>'$ is global. In terms of matrix orders, if $>'$ is given by a matrix M', then $>$ is given by the matrix $M = -M'$.

Example 9.6. The **negative lexicographic order** $>_{\texttt{ls}}$ is defined to be the inverse of $>_{\texttt{lp}}$ (the `s` in `ls` stands for "power series ring"). □

From a computational point of view, the most important examples of local monomial orders are obtained by considering **degree anticompatible orders** (instead of degree compatible orders in the global case).

Example 9.7. (1) The **negative degree reverse lexicographic order** $>_{\mathtt{ds}}$ is defined by setting

$$\boldsymbol{x}^\alpha >_{\mathtt{ds}} \boldsymbol{x}^\beta \;:\Longleftrightarrow \deg \boldsymbol{x}^\alpha < \deg \boldsymbol{x}^\beta \text{ or } (\deg \boldsymbol{x}^\alpha = \deg \boldsymbol{x}^\beta \text{ and the last nonzero entry of } \alpha - \beta \text{ is negative}).$$

(2) Let $\boldsymbol{w} = (w_1, \ldots, w_n) \in \mathbb{R}^n_{\geq 0}$ be a degree vector. The **weighted negative degree reverse lexicographic order** $>_{\mathtt{ws}(\boldsymbol{w})}$ is defined in the same way as $>_{\mathtt{ds}}$, with $\deg(\boldsymbol{x}^\alpha)$ being replaced by the $\boldsymbol{w}$-weighted degree $\boldsymbol{w}\text{-}\deg(\boldsymbol{x}^\alpha)$. [1] □

Note that further examples of local monomial orders are obtained by combining local monomial orders already known to product orders.

Mixed Monomial Orders

According to our definition in Lecture 1, a monomial order on $K[\boldsymbol{x}]$ is **mixed** if it is neither local nor global. In this case, we have strict inclusions

$$K[\boldsymbol{x}] \subsetneq K[\boldsymbol{x}]_> \subsetneq K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle}\,.$$

Example 9.8. Let $>_1$ be a monomial order on $K[\boldsymbol{x}]$, let $>_2$ be a monomial order on $K[\boldsymbol{y}]$, and let $>$ be the product order $(>_1, >_2)$ on $K[\boldsymbol{x}, \boldsymbol{y}]$.

(1) If $>_1$ is global and $>_2$ is local, then $S_> \subset K[\boldsymbol{x}, \boldsymbol{y}]$ consists precisely of the polynomials of type $a + f$, where $a \in K \setminus \{0\}$ and $f \in \langle \boldsymbol{y} \rangle \subset K[\boldsymbol{y}]$. Thus,

$$K[\boldsymbol{x}, \boldsymbol{y}]_> = (K[\boldsymbol{y}]_{\langle \boldsymbol{y} \rangle})[\boldsymbol{x}] = K[\boldsymbol{y}]_{\langle \boldsymbol{y} \rangle} \otimes_K K[\boldsymbol{x}]\,.$$

(2) If $>_1$ is local and $>_2$ is global, then $S_>$ consists precisely of the polynomials of type $a + f$, where $a \in K \setminus \{0\}$ and $f \in \langle \boldsymbol{x} \rangle \subset K[\boldsymbol{x}, \boldsymbol{y}]$. Thus,

$$(K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle})[\boldsymbol{y}] \subsetneq K[\boldsymbol{x}, \boldsymbol{y}]_> \subsetneq K[\boldsymbol{x}, \boldsymbol{y}]_{\langle \boldsymbol{x} \rangle}\,.$$ □

Remark 9.9. (1) In the situation of the example, if $>_1$ is global and $>_2$ is arbitrary, then $K[\boldsymbol{x}, \boldsymbol{y}]_> = (K[\boldsymbol{y}]_{>_2})[\boldsymbol{x}]$.
(2) To implement the ring $K[\boldsymbol{x}, \boldsymbol{y}]_{\langle \boldsymbol{x} \rangle}$, choose a local monomial order $>$ on $K(\boldsymbol{y})[\boldsymbol{x}]$ (considering $K(\boldsymbol{y})$ as the coefficient field). Then

$$K(\boldsymbol{y})[\boldsymbol{x}]_> = K[\boldsymbol{x}, \boldsymbol{y}]_{\langle \boldsymbol{x} \rangle}\,.$$ □

In Lecture 5, Example 5.15, we already made use of mixed orders to implement the ring $\mathbb{Q}[\boldsymbol{t}]_{\langle \boldsymbol{t} \rangle}[\boldsymbol{x}]$ in the context of studying flat families. Another important application of mixed orders is the elimination of variables in the local case (see Section 9.5 later in this lecture).

[1] Note that the `SINGULAR` implementation of a weighted negative degree reverse lexicographic order requires additionally that $w_1 > 0$.

Remark-Definition 9.10 (Monomial Orders on Free $\boldsymbol{K}[\boldsymbol{x}]_>$-Modules). Let $>$ be a monomial order on $K[\boldsymbol{x}]$, and let $F_>$ be a free $K[\boldsymbol{x}]_>$-module with a fixed basis $e_1, \dots, e_s$. Then a **monomial** in $F_>$ is an element of the form $\boldsymbol{x}^\alpha e_i$ and a **monomial order on $\boldsymbol{F}_>$** is nothing but a monomial order on the free $K[\boldsymbol{x}]$-module $F \subset F_>$ with basis $e_1, \dots, e_s$ (see Lecture 1, Remark-Definition 1.30). We usually require additionally that the monomial order on $F_>$ **extends** $>$, that is, that it induces $>$ on $K[\boldsymbol{x}]$. Note that there are many ways of extending $>$ to a monomial order on $F_>$ (see Lecture 3, Section 3.2.1.E). □

Ring Maps

Generalizing what we did in Lecture 3, we consider ring maps (that is, ring homomorphisms) between rings implemented by arbitrary monomial orders. In `SINGULAR`, these maps are created using the type `map` (or the commands `fetch` and `imap`) introduced in Section 3.2.2. Implementing a `map` means to choose images for the variables in the preimage ring. In the nonglobal case, this has to be done with some care. If $>_1$ and $>_2$ are monomial orders on $K[\boldsymbol{x}]$ and $K[\boldsymbol{y}] = K[y_1, \dots, y_m]$, each choice of elements $f_1, \dots, f_m \in K[\boldsymbol{x}]$ defines a ring map $K[\boldsymbol{y}] \to K[\boldsymbol{x}]_{>_1}$. In the nonglobal case, however, this does not necessarily induce a ring map $K[\boldsymbol{y}]_{>_2} \to K[\boldsymbol{x}]_{>_1}$. In fact, for this, the universal property of localization requires that the elements of $S_{>_2} \subset K[\boldsymbol{y}]$ are sent to units in $K[\boldsymbol{x}]_{>_1}$:

Lemma 9.11. *Let $>_1$ and $>_2$ be monomial orders on $K[\boldsymbol{x}]$ and $K[\boldsymbol{y}]$, and let $f_1, \dots, f_m \in K[\boldsymbol{x}]$. There is a ring map $K[\boldsymbol{y}]_{>_2} \to K[\boldsymbol{x}]_{>_1}$ taking the y_i to the f_i iff $h(f_1, \dots, f_m) \in S_{>_1} \subset K[\boldsymbol{x}]$ for each $h \in S_{>_2} \subset K[\boldsymbol{y}]$.*

If $>_1$ and $>_2$ are both local orders, the condition of Lemma 9.11 is equivalent to $f_1, \dots, f_m \in \langle \boldsymbol{x} \rangle \subset K[\boldsymbol{x}]$, that is, $f_1, \dots, f_m \notin S_{>_1}$. For mixed orders, however, the analogous condition ($f_i \notin S_{>_1}$ for each i such that $1 >_2 y_i$) is not sufficient. To give an example, consider the matrix orders $>_1$ and $>_2$ on $K[v,w]$ given by the integer matrices $\left(\begin{smallmatrix} -2 & 3 \\ 0 & -1 \end{smallmatrix}\right)$ and $\left(\begin{smallmatrix} -2 & 1 \\ 0 & -1 \end{smallmatrix}\right)$. Then $w >_1 1 >_1 v$ and $w >_2 1 >_2 v$, so both orders are mixed. Since $1 - vw \in S_{>_2}$ but $1 - vw \notin S_{>_1}$, the identity on $K[v,w]$ does not induce a ring map $K[v,w]_{>_2} \to K[v,w]_{>_1}$.

Warning. When creating a `map` (or applying `imap` or `fetch`) in `SINGULAR`, the condition given in Lemma 9.11 is not checked by the system:

```
> ring R = 0, (v,w), M(-2,1,0,-1);
> poly f = 1-vw;
> reduce(1,std(ideal(f))); // output shows: f is a unit in R
0
> ring S = 0, (v,w), M(-2,3,0,-1);
> map phi = R,v,w;          // implementing a nonexisting ring map
> reduce(1,std(phi(f)));    // output shows: f is not a unit in S
1
```

□

9.2 Standard Bases and their Computation

If $>$ is a local monomial order on $K[\boldsymbol{x}]$, every nonempty set of monomials in $K[\boldsymbol{x}]$ has a largest element with respect to $>$ (indeed, by inverting $>$, we obtain a well-order). In particular, we can define the **leading term** of a power series $f \in K[[\boldsymbol{x}]]$ with respect to $>$ as the largest term of f. If $>$ is a mixed monomial order on $K[\boldsymbol{x}]$, however, an arbitrary power series in $K[[\boldsymbol{x}]]$ may not have a largest term with respect to $>$. The situation is different for elements of $K[\boldsymbol{x}]_>$:

Remark-Definition 9.12. Let $>$ be any monomial order on $K[\boldsymbol{x}]$. If $0 \neq f \in K[\boldsymbol{x}]_>$, there is an element $u \in K[\boldsymbol{x}]$ such that $\mathrm{L}_>(u) = 1$ and $uf \in K[\boldsymbol{x}]$. The largest term of uf is independent of the choice of u and equals the largest term of the power series expansion of f. It is called the **leading term** of f, written $\mathrm{L}(f) = \mathrm{L}_>(f)$. If $\mathrm{L}(f) = a\boldsymbol{x}^\alpha$, with $a \in K$, then a is called the **leading coefficient** and $\boldsymbol{x}^\alpha$ the **leading monomial** of f. We refer to $f - \mathrm{L}(f)$ as the **tail** of f. Further, we set $\mathrm{L}(0) = \mathrm{L}_>(0) = 0$. If $I \subset K[\boldsymbol{x}]_>$ is an ideal, the **leading ideal** of I with respect to $>$ is defined to be the monomial ideal

$$\mathrm{L}_>(I) := \mathrm{L}(I) := \langle \mathrm{L}(f) \mid f \in I \rangle \subset K[\boldsymbol{x}]\,.$$

The monomials not in $\mathrm{L}_>(I)$ are called **standard monomials** (for I, with respect to $>$). $\square$

Example 9.13. In the case of one variable x, there is precisely one local monomial order: $1 > x > x^2 > \cdots$. With respect to this order, if

$$f = \frac{2x}{1-x} + x = 3x + \sum_{k=2}^{\infty} 2x^k \in K[x]_{\langle x \rangle}\,,$$

then $\mathrm{L}(f) = \mathrm{L}((1-x)f) = 3x \in K[x]$. $\square$

Definition 9.14. Let $>$ be a monomial order on $K[\boldsymbol{x}]$, and let $I \subset K[\boldsymbol{x}]_>$ be an ideal. A finite subset $\{f_1, \dots, f_r\} \subset I$ is called a **standard basis** for I with respect to $>$ if

$$\mathrm{L}_>(I) = \langle \mathrm{L}_>(f_1), \dots, \mathrm{L}_>(f_r) \rangle \subset K[\boldsymbol{x}]\,.$$

We say that a finite subset $\mathcal{G}$ of $K[\boldsymbol{x}]_>$ is a **standard basis over $\boldsymbol{K[x]_>}$** if it is a standard basis for the ideal of $K[\boldsymbol{x}]_>$ generated by $\mathcal{G}$. $\square$

Remark 9.15. As for Gröbner bases, the existence of a standard basis for a given ideal $I \subset K[\boldsymbol{x}]_>$ is guaranteed by Gordan's lemma. Moreover, the Mora Division Theorem 9.19 below implies that each standard basis for I is a generating set for I. Finally, the concept of standard bases extends to free $K[\boldsymbol{x}]_>$-modules $F_>$ with a fixed basis $e_1, \dots, e_s$. $\square$

The definition of **standard bases over $\boldsymbol{K}[[\boldsymbol{x}]]$** (with respect to a local monomial order) is completely analogous to Definition 9.14. Further, Remark 9.15 applies accordingly (replace the Mora Division Theorem by the Grauert Division Theorem 9.16 below).[2]

Division with Remainder and Normal Forms

A nonzero power series $g \in K[[\boldsymbol{x}]] = K[[x_1, \dots, x_n]]$ is called **$\boldsymbol{x_n}$-general of order $\boldsymbol{m}$** if its leading monomial with respect to $>_{1\mathrm{s}}$ equals x_n^m. The classical Weierstrass division theorem for formal power series asserts that if $f, f_1 \in K[[\boldsymbol{x}]]$ are power series such that f_1 is x_n-general of order m, then there exist uniquely determined $g_1 \in K[[\boldsymbol{x}]]$ and $h \in K[[x_1, \dots, x_{n-1}]][x_n]$ such that

$$f \;=\; g_1 f_1 + h, \qquad \deg_{x_n}(h) \leq m - 1\,.$$

See Grauert and Remmert (1971), §I.4, Satz 2 for a proof.

Extending the Weierstrass division theorem, the following theorem treats the simultaneous division by several power series. More generally, in analogy to the Division with Remainder Theorem 1.38 for polynomial vectors in Lecture 1, we may divide in free $K[[\boldsymbol{x}]]$-modules:

Theorem 9.16 (Grauert Division Theorem). *Let $\widehat{F}$ be a free $K[[\boldsymbol{x}]]$-module with a fixed basis $e_1, \dots, e_s$, let $>$ be a local monomial order on $\widehat{F}$, and let $f_1, \dots, f_r \in \widehat{F} \setminus \{0\}$. For every $f \in \widehat{F}$, there exist $g_1, \dots, g_r \in K[[\boldsymbol{x}]]$ and an element $h \in \widehat{F}$ such that*

$$f = \sum_{k=1}^{r} g_k f_k + h\,,$$

where:

(DIV 1) $\mathrm{L}(f) \geq \mathrm{L}(g_k f_k)$ *whenever both sides are nonzero.*
(DIV 2) If h is nonzero, no term of h is divisible by any $\mathrm{L}(f_k)$.

See Decker and Schreyer (2006) for a determinate version of the theorem and its proof.

Remark 9.17. (1) As in the polynomial case, we also have the following condition which is weaker than (DIV 2):

(DIV 2') If $h \neq 0$, then $\mathrm{L}(h)$ *is not divisible by any* $\mathrm{L}(f_k)$.

We call each expression $f = \sum_{k=1}^{r} g_k f_k + h$ satisfying (DIV 1) and (DIV 2') a **standard expression** for f in terms of the f_k with **remainder** h. If all components of the vectors $f, f_1, \dots, f_r$ are in $K[\boldsymbol{x}]$, a standard expression is called **polynomial** if the factors g_k and the components of the remainder h are polynomials in $K[\boldsymbol{x}]$, too.

[2] If $\widehat{F}$ is a free $K[[\boldsymbol{x}]]$-module with a fixed basis $e_1, \dots, e_s$, then a **local monomial order on $\widehat{\boldsymbol{F}}$** is nothing but a local monomial order on the free $K[\boldsymbol{x}]$-module $F \subset \widehat{F}$ with basis $e_1, \dots, e_s$.

(2) Lecture 1, Remark 1.40 on normal forms applies accordingly.

(3) Standard bases over $K[[\boldsymbol{x}]]$ are characterized by **Buchberger's criterion over $\boldsymbol{K}[[\boldsymbol{x}]]$**, which is based on the Grauert division theorem, and which reads word for word identically to Buchberger's criterion in the polynomial case. □

The Grauert division theorem and, as a consequence, Buchberger 's criterion over $K[[\boldsymbol{x}]]$ are not of practical interest. Even when starting with *polynomials* $f, f_1, \ldots, f_r \in K[\boldsymbol{x}]$, the Grauert division theorem only asserts that the remainder h and the factors $g_1, \ldots, g_r$ exist as formal power series. In fact, a *polynomial* standard expression for f in terms of the f_k may not exist. Further, if $n \geq 2$, there is no analogue to the Grauert division theorem which holds over $K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle}$ (the condition (DIV 2) cannot always be achieved).

Example 9.18. (1) Applying the division process from Lecture 1 with respect to the unique local monomial order on $K[x]$, it takes infinitely many steps to divide $f = x$ by $f_1 = x - x^2$. As a result, we get the standard expression

$$f = g_1 f_1 + 0, \quad \text{where } \; g_1 = \frac{1}{1-x} = \sum_{k=0}^{\infty} x^k \,. \tag{9.1}$$

This standard expression is uniquely determined by f, f_1 and is not polynomial.

(2) Consider the polynomials $f = y$ and $f_1 = y - y^2 - x$ as elements of the local ring $K[x,y]_{\langle x,y \rangle} \subset K[[x,y]]$, and fix the negative lexicographic order $>_{\mathtt{ls}}$ on $K[x,y]$ (with $y >_{\mathtt{ls}} x$). Suppose that there is a standard expression $f = g_1 f_1 + h$ satisfying (DIV 2), where $g_1, h \in K[x,y]_{\langle x,y \rangle}$. Then each term of the remainder h is not divisible by $\mathrm{L}(f_1) = y$, that is, $h \in K[x]_{\langle x \rangle}$. This implies that $y = \mathrm{L}(f) = \mathrm{L}(g_1 f_1) = \mathrm{L}(g_1) \cdot y$ and, thus, that g_1 is a unit in $K[x,y]_{\langle x,y \rangle}$. Further, substituting h for y in $f = g_1 f_1 + h$, we get the equality

$$g_1(x,h) \cdot (h - h^2 - x) = 0 \in K[x]_{\langle x \rangle} \,.$$

Since f and f_1 vanish at the origin, h cannot have a constant term. It follows that $g_1(x,h) \neq 0$ since g_1 is a unit. We conclude that $h - h^2 - x = 0$. Writing h as $\frac{h_1}{1+h_2}$, with $h_1 \in K[x]$ and $h_2 \in \langle x \rangle$, we get the equality

$$(1 + h_2) \cdot h_1 - h_1^2 - x \cdot (1 + h_2)^2 = 0 \in K[x] \,.$$

A check on degrees shows that this is impossible. □

It is clear from the first example above that for computations with local (and mixed) orders, we have to modify the division process. Inspecting the standard expression (9.1), we see what should be done. Namely, by rewriting (9.1), we get a polynomial standard expression for $(1 - x) \cdot f$:

$$(1 - x) \cdot f = 1 \cdot f_1 + 0 \,.$$

Similarly, in the second example, we get a polynomial standard expression for $(1-y)\cdot f$:

$$(1-y)\cdot f = 1\cdot f_1 + x$$

(the resulting standard expression

$$f = \frac{1}{1-y}\cdot f_1 + \frac{x}{1-y}$$

for f in $K[x,y]_{\langle x,y\rangle}$ does, of course, not satisfy (DIV 2)). Note that in both cases, we obtained a polynomial standard expression for f times a polynomial unit in the ring $K[\boldsymbol{x}]_{\langle \boldsymbol{x}\rangle}$. In general, we have the following result:

Theorem 9.19 (Mora Division Theorem). *Let F be a free $K[\boldsymbol{x}]$-module with a fixed basis $e_1,\dots,e_s$, let $>$ be a monomial order on F, and let $f_1,\dots,f_r \in F\setminus\{0\}$. For every $f\in F$, there exist polynomials $u, g_1,\dots,g_r \in K[\boldsymbol{x}]$ and an element $h\in F$ such that*

$$u\cdot f = \sum_{k=1}^{r} g_k f_k + h, \qquad \mathrm{L}(u) = 1\,,$$

and:

(DIV 1) $\mathrm{L}(f) \geq \mathrm{L}(g_k f_k)$ *whenever both sides are nonzero.*
(DIV 2') If h is nonzero, then $\mathrm{L}(h)$ *is not divisible by any* $\mathrm{L}(f_k)$.

Equivalently, we may formulate the Mora division theorem as a division theorem over the ring $K[\boldsymbol{x}]_>$ implemented by the monomial order $>$:

Corollary 9.20. *Let $>$ be a monomial order on $K[\boldsymbol{x}]$, let $F_>$ be a free $K[\boldsymbol{x}]_>$-module with a fixed basis $e_1,\dots,e_s$ and with a monomial order extending $>$, and let $f_1,\dots,f_r \in F_>\setminus\{0\}$ be polynomial vectors. For every $f\in F_>$, there exist elements $g_1,\dots,g_r\in K[\boldsymbol{x}]_>$ and an element $h\in F_>$ such that*

$$f = \sum_{k=1}^{r} g_k f_k + h$$

and:

(DIV 1) $\mathrm{L}(f) \geq \mathrm{L}(g_k f_k)$ *whenever both sides are nonzero.*
(DIV 2') If h is nonzero, then $\mathrm{L}(h)$ *is not divisible by any* $\mathrm{L}(f_k)$.

We prove the Mora division theorem by presenting a general indeterminate **division algorithm** (due to Greuel and Pfister, based on ideas of Mora and Lazard). In comparison to the global case, there is one new feature — we are not only dividing by $f_1,\dots,f_r$, but also by some of the intermediate dividends. To decide whether an intermediate dividend h should be stored as a possible divisor for division steps still to come, the ecart of h is computed.

Definition 9.21. Let F be a free $K[\boldsymbol{x}]$-module with a fixed basis $e_1, \dots, e_s$, and let $>$ be a monomial order on F. If $a\boldsymbol{x}^\alpha e_i$ is a nonzero term in F, we set

$$\deg \boldsymbol{x}^\alpha e_i := \deg \boldsymbol{x}^\alpha .$$

If $f \in F \setminus \{0\}$, the **ecart** of f (with respect to $>$) is defined to be

$$\operatorname{ecart}(f) := \deg f - \deg \mathrm{L}(f) ,$$

where $\deg f$ is the maximum degree of a term of f. □

Algorithm 9.22 (Mora Division).

Let F be a free $K[\boldsymbol{x}]$-module with a fixed basis $e_1, \dots, e_s$, and let $>$ be a monomial order on F.

```
INPUT:  $f \in F$, $f_1, \dots, f_r \in F \setminus \{0\}$.
OUTPUT: the remainder $h \in F$ in a polynomial standard expression for
        $u \cdot f$ in terms of $f_1, \dots, f_r$, where $u$ is a polynomial unit in
        $K[\boldsymbol{x}]_>$.
```

```
Step 1. Set $h := f$ and $D := \{f_1, \dots, f_r\}$;
Step 2. while $\big(h \neq 0$ and $D_h := \{g \in D \mid \mathrm{L}(g)$ divides $\mathrm{L}(h)\} \neq \emptyset\big)$
          • choose $g \in D_h$ with $\operatorname{ecart}(g)$ minimal;
          • if $(\operatorname{ecart}(g) > \operatorname{ecart}(h))$ then $D := D \cup \{h\}$;
          • set $h := h - \frac{\mathrm{L}(h)}{\mathrm{L}(g)} g$;
Step 3. return($h$).
```
□

Note that in a single division step, h is added to D iff $\mathrm{L}(h) \in \langle \mathrm{L}(g) \mid g \in D \rangle$, but $t^{\operatorname{ecart}(h)}\mathrm{L}(h) \notin T := \langle t^{\operatorname{ecart}(g)}\mathrm{L}(g) \mid g \in D \rangle \subset F \otimes_{K[\boldsymbol{x}]} K[t, \boldsymbol{x}]$, where t is an extra variable. Due to Gordan's lemma, the submodule T can only be enlarged finitely many times in the process. Thus, to show termination of the Mora division algorithm, we may assume that T is not enlarged in the process. Then Algorithm 9.22 performs the same steps as the division with remainder algorithm discussed in Lecture 1 applied to the homogenizations[3] $f^{\mathrm{hom}}, f_1^{\mathrm{hom}}, \dots, f_r^{\mathrm{hom}}$ and the global monomial order defined by

$$\begin{aligned} t^c \boldsymbol{x}^\alpha e_i >_t t^d \boldsymbol{x}^\beta e_j :\Longleftrightarrow {} & c + |\alpha| > d + |\beta| ,\ \text{or} \\ & \big(c + |\alpha| = d + |\beta| \text{ and } \boldsymbol{x}^\alpha e_i > \boldsymbol{x}^\beta e_j\big) \end{aligned}$$

(note that $>_t$ is defined such that $\mathrm{L}_{>_t}(g^{\mathrm{hom}}) = t^{\operatorname{ecart}(g)}\mathrm{L}_>(g)$). Thus, termination follows from the termination result in Lecture 1.

To show the correctness of the algorithm is a simple exercise. See Greuel and Pfister (2002) for details.

[3] Here, the **homogenization** g^{hom} of a vector $g = \sum_{\alpha,i} c_\alpha \boldsymbol{x}^\alpha e_i \in F$ is defined as

$$g^{\mathrm{hom}} = \sum_{\alpha,i} c_\alpha t^{\deg(g) - |\alpha|} \boldsymbol{x}^\alpha e_i \in F \otimes_{K[\boldsymbol{x}]} K[t, \boldsymbol{x}] .$$

Remark 9.23. (1) If we apply the Mora algorithm to homogeneous polynomials $f, f_1, \ldots, f_r$, the ecart is always zero, and we obtain the indeterminate division algorithm described in Lecture 1.

(2) If $>$ is a global monomial order, then $\mathrm{L}(g) \mid \mathrm{L}(h)$ implies that $\mathrm{L}(g) \leq \mathrm{L}(h)$. Hence, even if added to D in the division process, h will not be used in further division steps. Thus, we obtain again the division algorithm of Lecture 1 (with the freedom of choice being reduced). □

A version of the Mora division algorithm is implemented in `SINGULAR` and accessible via the `reduce` and `division` commands. If called by the `reduce` command, only the remainder h is returned. In contrast, the `division` command also yields the unit u and the quotients g_k (this requires some extra bookkeeping).

Computing Standard Bases

Based on Mora's division theorem, we get Buchberger's criterion over $K[\boldsymbol{x}]_>$. Having practical computations in mind, we formulate the criterion such that it involves polynomial data only:

Theorem 9.24 (Buchberger's Criterion). *Let $>$ be a monomial order on $K[\boldsymbol{x}]$, let $F_>$ be a free $K[\boldsymbol{x}]_>$-module with a fixed basis $e_1, \ldots, e_s$ and with a monomial order extending $>$. Moreover, let $f_1, \ldots, f_r \in F_> \setminus \{0\}$ be polynomial vectors. Compute for each pair of indices $i > j$ a polynomial standard expression*

$$u^{(ij)} \cdot \mathrm{S}(f_i, f_j) = \sum_{k=1}^{r} g_k^{(ij)} f_k + h_{ij}$$

such that $u^{(ij)} \in K[\boldsymbol{x}]$ is a polynomial with a constant leading term. Then $f_1, \ldots, f_r$ form a standard basis over $K[\boldsymbol{x}]_>$ iff all remainders h_{ij} are zero.

Remark 9.25. (1) As in the special case of global monomial orders, Buchberger's criterion yields algorithms for computing standard bases and syzygies over $K[\boldsymbol{x}]_>$ (working with polynomial data). In theoretical terms, we can make use of Schreyer's syzygy algorithm to extend Hilbert's syzygy theorem to the ring $K[\boldsymbol{x}]_>$ (see Greuel and Pfister (2002), Section 2.5).

(2) Lecture 2, Remark 2.9 on the role of the coefficient field applies accordingly. □

Remark 9.26. Let $>$ be a local monomial order on $K[\boldsymbol{x}]$, and let $f_1, \ldots, f_r \in K[\boldsymbol{x}]^s$ be polynomial vectors. It follows from the two versions of Buchberger's criterion considered in this section that $\{f_1, \ldots, f_r\}$ is a standard basis over $K[\boldsymbol{x}]_>$ iff it is a standard basis over $K[[\boldsymbol{x}]]$. □

9.3 Factorization and Primary Decomposition

Let $>$ be a monomial order on $K[\boldsymbol{x}]$, and let $f \in K[\boldsymbol{x}]$ be a nonzero polynomial. Then we may ask for finding the irreducible factors of f in the localized ring $K[\boldsymbol{x}]_>$. We already know from Remark 7.1 in Lecture 7 that in SINGULAR, if K is a finite field, the field $\mathbb{Q}$, or a number field, the factorization of f in $K[\boldsymbol{x}]$ can be computed using the `factorize` command. If $f = c \cdot f_1^{m_1} \cdot \ldots \cdot f_s^{m_s}$ is this factorization, the irreducible factors of f in $K[\boldsymbol{x}]_>$ are precisely the f_i which are non-units in $K[\boldsymbol{x}]_>$:

$$f = \underbrace{c \cdot \prod_{\mathrm{L}(f_i) \in K} f_i^{m_i}}_{\text{unit}} \cdot \prod_{\mathrm{L}(f_i) \notin K} f_i^{m_i}$$

is the factorization of f into irreducibles in $K[\boldsymbol{x}]_>$.

Example 9.27. We use SINGULAR to factorize a bivariate polynomial of degree 19 over $\mathbb{Q}$:

```
> ring R = 0, (x,y), ds;   // local monomial order
> poly f = y13-y15+x2y9+x2y10-2x2y11-x2y12+x2y13-3x3y11+3x3y13+x4y6
. -x4y7-2x4y8+x4y9+x4y10-3x5y7-3x5y8+5x5y9-x5y10-2x5y11+4x5y12-x6y4
. +x6y6+3x6y9-3x6y11-3x7y4+2x7y5+x7y6-6x7y7+6x7y8+3x7y9-4x7y10+x7y11
. +3x8y5+3x8y6-4x8y7+x8y8+x8y9-4x8y10+2x9y2-4x9y3+2x9y4+7x9y5-5x9y6
. -3x9y7+x9y8+3x10y2-x10y3+5x10y5-3x10y6-3x10y7-x10y9+4x11y-x11y2
. -4x11y3+x11y4-x12+4x12y+x12y2-3x12y3+x12y4-x12y5-x12y6+x14-x14y2;
> factorize(f);
[1]:
   _[1]=-1
   _[2]=-y4+2x3y2-x6+4x5y+x7
   _[3]=x2-y2+x3
   _[4]=x2+y3
   _[5]=x2+y4
   _[6]=-1+y
   _[7]=1+y
[2]:
   1,1,1,1,1,1,1
```

Though we have chosen an order which implements $\mathbb{Q}[x,y]_{\langle x,y\rangle}$, the `factorize` command returns the irreducible factors in $\mathbb{Q}[x,y]$ (it does not single out the factors which are irreducible in $\mathbb{Q}[x,y]_{\langle x,y\rangle}$). A check on the leading terms shows that the factors $-1+y$ and $1+y$ (and the constant -1) are units in $\mathbb{Q}[x,y]_{\langle x,y\rangle}$, while the other factors are not. Hence, up to the unit $u = 1-y^2$, the factorization of f into irreducibles in $\mathbb{Q}[x,y]_{\langle x,y\rangle}$ is the product

$$(-y^4 + 2x^3y^2 - x^6 + 4x^5y + x^7) \cdot (x^2 - y^2 + x^3) \cdot (x^2 + y^3) \cdot (x^2 + y^4)\,.$$

The rightmost factor decomposes over the extension field $\mathbb{Q}(i)$ as $x^2 + y^4 = -(ix+y^2)(ix-y^2)$. The other factors can be shown to be absolutely irreducible (for instance, use `absFactorize`). □

Remark 9.28. A similar recipe works for primary decomposition: if I is an ideal of $K[\boldsymbol{x}]$, and if $I = \bigcap_{i=1}^{s} Q_i$ is a primary decomposition of I, then

$$I\,K[\boldsymbol{x}]_> = \bigcap_{Q_i \subset K[\boldsymbol{x}] \setminus S_>} Q_i\,K[\boldsymbol{x}]_>$$

is a primary decomposition of $I\,K[\boldsymbol{x}]_>$. Note that, if $>$ is a local order, then $Q_i \subset K[\boldsymbol{x}] \setminus S_>$ iff none of the given generators for Q_i has a constant leading term with respect to $>$. □

We may also ask for the factorization of f in the power series ring $K[[\boldsymbol{x}]]$.[4] Here, the situation is different. The only case in which the factorization problem in a power series ring can be tackled successfully is the case of (at most) two variables. We will discuss this in Section 9.6.

9.4 Computing Dimension

Let $>$ be a monomial order on $K[\boldsymbol{x}]$. We already know that if $>$ is global, then $\dim\bigl(K[\boldsymbol{x}]/I\bigr) = \dim\bigl(K[\boldsymbol{x}]/\mathrm{L}_>(I)\bigr)$ for each ideal $I \subset K[\boldsymbol{x}]$. Indeed, if I is homogeneous, this follows from Macaulay's Theorem 1.35 on Hilbert functions, while for I arbitrary, the statement is obtained as a corollary of Theorem 5.16 on Flatness and Gröbner Bases (see Remark 5.18).

Starting from a version of Theorem 5.16 which holds for arbitrary monomial orders, we obtain, more generally, the first statement of the theorem below. The second statement, which is also a consequence of the extended version of Theorem 5.16, generalizes Macaulay's Theorem 1.33 on standard monomials. See Greuel and Pfister (2002), Section 7.5 for details.

Theorem 9.29. *Let $>$ be any monomial order on $K[\boldsymbol{x}]$, and let $I \subset K[\boldsymbol{x}]_>$ be an ideal. Then we have:*

(1) $\dim(K[\boldsymbol{x}]_>/I) = \dim(K[\boldsymbol{x}]/\mathrm{L}_>(I))$.
(2) $\dim_K(K[\boldsymbol{x}]_>/I) = \dim_K(K[\boldsymbol{x}]/\mathrm{L}_>(I))$.

Moreover, if $>$ is global or if $\dim_K(K[\boldsymbol{x}]_>/I) < \infty$, then the standard monomials represent a K-basis for $K[\boldsymbol{x}]_>/I$.

Note that the standard monomials are always K-linearly independent modulo I. If $>$ is not global and $\dim_K(K[\boldsymbol{x}]_>/I) = \infty$, their residue classes may fail, however, to generate the quotient ring. For instance, $\frac{1}{1-x} = \sum_{k=0}^{\infty} x^k \in K[x]_{\langle x\rangle}$ cannot be written as a finite K-linear combination of the x^k. On the other hand, the x^k are precisely the standard monomials for the zero ideal of $K[x]_{\langle x\rangle}$ (with respect to the unique local monomial order on $K[x]$).

[4] If $K \subset \mathbb{C}$, we can deduce from Artin's approximation theorem that the factorization in $K\{\boldsymbol{x}\}$ coincides with that in $K[[\boldsymbol{x}]]$. See de Jong and Pfister (2000), Thm. 8.1.1 for Artin's approximation theorem.

Remark 9.30 (The `SINGULAR` Commands `dim` and `vdim`). Theorem 9.29 and Buchberger's algorithm in its general form as discussed in Section 9.2 reduce the computation of dimension to a problem concerning monomial ideals of $K[\boldsymbol{x}]$ which is of a purely combinatorial nature (as already explained in Lecture 6, Section 6.1.1). Namely, if $>$ is a monomial order on $K[\boldsymbol{x}]$, if $I \subset K[\boldsymbol{x}]_>$ is an ideal, and if we have computed a standard basis $\mathcal{G}$ for I, then the Krull dimension of $K[\boldsymbol{x}]_>/I$ is the maximal cardinality of a subset $\boldsymbol{u} \subset \{x_1, \dots, x_n\}$ such that no leading term of an element of $\mathcal{G}$ is in $K[\boldsymbol{u}]$. The K-vector space dimension of $K[\boldsymbol{x}]_>/I$ equals, then, the number of monomials $\boldsymbol{x}^\alpha$ such that no leading term of an element of $\mathcal{G}$ divides $\boldsymbol{x}^\alpha$. These numbers can be computed using the `SINGULAR` commands `dim` and `vdim`, respectively.[5] □

Given an ideal $I \subset K[\boldsymbol{x}]$, we may use `dim` to compute the dimension of the algebraic set $A = \mathrm{V}(I) \subset \mathbb{A}^n = \mathbb{A}^n(\overline{K})$ (the **global dimension**), and we may use it to compute the dimension of A at a given point $q \in A$ (the **local dimension** at q). Indeed, according to Lecture 6, Section 6.1.1, we have

$$\dim A = \dim \overline{K}[\boldsymbol{x}]/\mathrm{I}(A) = \dim K[\boldsymbol{x}]/I\,,$$

and this number can be computed choosing a global monomial order on $K[\boldsymbol{x}]$. Further, if $q = p$ is the origin, then

$$\dim_p A = \dim K[\boldsymbol{x}]_{\langle \boldsymbol{x}\rangle}/I\,K[\boldsymbol{x}]_{\langle \boldsymbol{x}\rangle}\,, \tag{9.2}$$

and this number can be computed choosing a local monomial order on $K[\boldsymbol{x}]$.

For (9.2), recall that, in general,

$$\dim_q A = \dim \mathcal{O}_{A,q} = \dim \overline{K}[\boldsymbol{x}]_{\mathfrak{m}_q}/I(A)\,\overline{K}[\boldsymbol{x}]_{\mathfrak{m}_q}\,,$$

where $\mathfrak{m}_q$ denotes the maximal ideal of $\overline{K}[\boldsymbol{x}]$ consisting of all polynomials in $\overline{K}[\boldsymbol{x}]$ vanishing at q (see Lecture 2, Remark 2.12). Thus, we may argue as in Lecture 6, Section 6.1.1 (applying Hilbert's Nullstellensatz and Theorem 9.29).

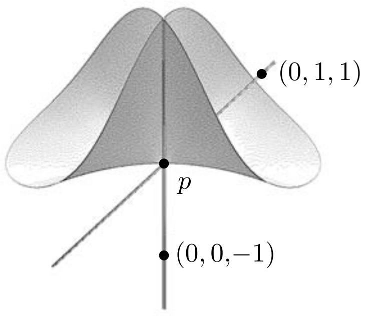
If q is different from the origin, then $\dim_q A$ can be computed by translating q to the origin as in the following example (if necessary, replace K by an appropriate extension field which contains all coordinate entries of q):

Example 9.31. We consider the ideal

$$I = \langle x(x^2 - y^2 z),\, (y - z)(x^2 - y^2 z)\rangle \subset \mathbb{Q}[x, y, z]\,,$$

the algebraic set $A = \mathrm{V}(I) \subset \mathbb{A}^3(\mathbb{C})$, and the three points $p = (0,0,0)$, $(0,1,1)$, and $(0,0,-1)$ on A:

[5] If the input for `dim` is not a standard basis, `SINGULAR` prints a warning and computes the dimension of the monomial ideal generated by the leading terms of the given generators. Similarly for `vdim`.



```
> ring R = 0, (x,y,z), dp;
> ideal I = x*(x2-y2z), (y-z)*(x2-y2z);
> ideal G = std(I); G;       // compute Groebner basis
G[1]=y3z-y2z2-x2y+x2z
G[2]=xy2z-x3
```

We see that $\boldsymbol{u} = \{x, y\}$ is a set of variables of maximal cardinality such that no leading term of G is in $\mathbb{Q}[\boldsymbol{u}]$. Hence, $\dim A = 2$:

```
> dim(G);                    // global dimension
2
> ring R1 = 0, (x,y,z), ds;  // implements local ring at (0,0,0)
> ideal I = imap(R,I);
> ideal G = std(I); G;       // compute standard basis
G[1]=x3-xy2z
G[2]=x2y-x2z-y3z+y2z2
> dim(G);                    // local dimension at (0,0,0)
2
> map phi = R,x,y-1,z-1;     // translate (0,1,1) to the origin
> I = phi(I);
> G = std(I); G;
G[1]=x
G[2]=y-z-2y2+yz+z2+x2y+y3-x2z+y2z-2yz2-y3z+y2z2
> dim(G);                    // local dimension at (0,1,1)
1
```

Pausing for a moment, we read from the output that

$$\dim \mathfrak{m} + \operatorname{codim} \mathfrak{m} = 0 + 1 \neq 2 = \dim \mathbb{Q}[x, y, z]/I\,,$$

where $\mathfrak{m} = \langle \overline{x}, \overline{y} - 1, \overline{z} - 1 \rangle \subset \mathbb{Q}[x, y, z]/I$ (indeed, $\mathfrak{m}$ is a maximal ideal and $\operatorname{codim} \mathfrak{m} = \dim(\mathbb{Q}[x, y, z]/I)_{\mathfrak{m}}$ is the dimension of A at $(0, 1, 1)$). This is an example of an ideal of an affine ring whose dimension and codimension do not add up to the dimension of the ring (see Lecture 2, Remark 2.10).

Now, we continue our session:

```
> map psi = R,x,y,z+1;       // translate (0,0,-1) to the origin
> I = psi(I);
> G = std(I);
> G;
G[1]=x2-y2-x2y+y3+x2z-2y2z+y3z-y2z2
> dim(G);                    // local dimension at (0,0,-1)
2
```

That the local dimension of A at $(0,0,-1)$ is 2 indicates that the real picture of A displayed on the previous page is misleading. In contrast to the line $\mathrm{V}(x, y-z)$, the line $\mathrm{V}(x,y)$ is not an irreducible component of A:

```
> setring R;
> LIB "primdec.lib";
> list L = minAssGTZ(G); L;
[1]:
   _[1]=y-z
   _[2]=x
[2]:
   _[1]=-y2z+x2
```

We refer to $\mathrm{V}(x^2 - y^2z)$ as the **Whitney umbrella**. The "stick" $\mathrm{V}(x,y)$ is its singular locus:

```
> ideal SLoc = L[2][1], jacob(L[2][1]);
> radical(SLoc);
_[1]=y
_[2]=x
```

$\square$

Let $I \subsetneq K[\boldsymbol{x}]$ be a proper ideal, and let $A = \mathrm{V}(I) \subset \mathbb{A}^n = \mathbb{A}^n(\overline{K})$. Recall from Theorem 6.1 in Lecture 6 that the global dimension of A is zero iff A is finite. Similarly, the local dimension of A at a point $q \in A$ is zero iff q is an **isolated point** of A, that is, if there is a Zariski open neighborhood U of q in $\mathbb{A}^n$ containing no other points of A. In fact, q is an isolated point of A iff $\mathfrak{m}_q$ is among the minimal associated primes of $I\overline{K}[\boldsymbol{x}]$ iff $I\overline{K}[\boldsymbol{x}]_{\mathfrak{m}_q}$ is $\mathfrak{m}_q\overline{K}[\boldsymbol{x}]_{\mathfrak{m}_q}$-primary iff $\overline{K}[\boldsymbol{x}]_{\mathfrak{m}_q}/I\overline{K}[\boldsymbol{x}]_{\mathfrak{m}_q}$ is a finite dimensional $\overline{K}$-vector space. Making use of this fact, we extend the notion of multiplicity defined for roots of univariate polynomials as follows:

Definition 9.32. Let $I \subset K[\boldsymbol{x}]$ be an ideal, and suppose that $q \in \mathrm{V}(I) \subset \mathbb{A}^n$ is an isolated point of $\mathrm{V}(I)$. The **multiplicity of *q* as a solution of *I*** is defined to be

$$\mathrm{mult}\,(q \mid I) = \dim_{\overline{K}} \overline{K}[\boldsymbol{x}]_{\mathfrak{m}_q}/I\overline{K}[\boldsymbol{x}]_{\mathfrak{m}_q}\,. \qquad \square$$

In particular, an isolated point $q \in \mathrm{V}(I)$ has multiplicity 1 as a solution of I iff $I\overline{K}[\boldsymbol{x}]_{\mathfrak{m}_q}$ equals $\mathfrak{m}_q\overline{K}[\boldsymbol{x}]_{\mathfrak{m}_q}$, that is, iff $I\overline{K}[\boldsymbol{x}]_{\mathfrak{m}_q}$ is radical.

Note that if $q = p$ is the origin, then

$$\operatorname{mult}(p \mid I) = \dim_K K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle} / I K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle} \,,$$

and this number can be computed using the `vdim` command. If q is different from the origin, translate it to the origin.

Remark 9.33. If $I \subset K[\boldsymbol{x}]$ is a zero-dimensional ideal, there is an isomorphism of rings

$$\overline{K}[\boldsymbol{x}]/I\overline{K}[\boldsymbol{x}] \cong \prod_{q \in \mathrm{V}(I)} \overline{K}[\boldsymbol{x}]_{\mathfrak{m}_q} / I\overline{K}[\boldsymbol{x}]_{\mathfrak{m}_q}$$

(see Cox, Little, and O'Shea (1998), Theorem 4.2.2). In particular, we have, then,

$$\dim_K K[\boldsymbol{x}]/I = \dim_{\overline{K}} \overline{K}[\boldsymbol{x}]/I\overline{K}[\boldsymbol{x}] = \sum_{q \in V(I)} \operatorname{mult}(q \mid I) \,. \tag{9.3}$$

□

If K is a perfect field, this formula allows us to read the multiplicities of the solutions from a primary decomposition of I:

Remark 9.34. Let $I \subset K[\boldsymbol{x}]$ be an ideal and suppose that Q is an isolated zero-dimensional primary component of I with radical P. Then P is a maximal ideal of $K[\boldsymbol{x}]$ and the points of $\mathrm{V}(P) \subset \mathbb{A}^n = \mathbb{A}^n(\overline{K})$ are pairwise conjugate over K. If K is a perfect field, each such point has multiplicity 1 as a solution of P. In this case, it follows from formula (9.3) that there are precisely $\dim_K K[\boldsymbol{x}]/P$ such points and if q is one of them, then

$$\operatorname{mult}(q \mid I) = \operatorname{mult}(q \mid Q) = \frac{\dim_K K[\boldsymbol{x}]/Q}{\dim_K K[\boldsymbol{x}]/P} \,.$$

In particular, $\operatorname{mult}(q \mid I) = 1$ iff $Q = P$. □

In what follows, we give two examples of computing multiplicities. The first example deals with Tjurina numbers:

Definition 9.35. Let $f \in K[\boldsymbol{x}]$. The **Tjurina number** of f at $q \in \mathbb{A}^n$ is defined to be

$$\tau_q(f) = \dim_{\overline{K}} \overline{K}[\boldsymbol{x}]_{\mathfrak{m}_q} / T_f \overline{K}[\boldsymbol{x}]_{\mathfrak{m}_q},$$

where $T_f \subset K[\boldsymbol{x}]$ is the **Tjurina ideal of *f***, that is, the ideal generated by f and its partial derivatives. In particular, the Tjurina number at the origin $p \in \mathbb{A}^n$ is

$$\begin{aligned}\tau_p(f) &= \dim_K K[\boldsymbol{x}]_{\langle \boldsymbol{x} \rangle} \Big/ \Big\langle f, \frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n} \Big\rangle \\ &= \dim_K K[[\boldsymbol{x}]] \Big/ \Big\langle f, \frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n} \Big\rangle \,.\end{aligned}$$

□

If the Tjurina ideal T_f is zero-dimensional and if $n \geq 2$, then f is square-free, $\mathrm{V}(T_f)$ is the singular locus A_{sing} of $A = \mathrm{V}(f) \subset \mathbb{A}^n$ (see Lecture 2, Theorem 2.20), and the Tjurina number $\tau_q(f)$ is the multiplicity of q as a solution of T_f. Further, (9.3) implies that

$$\dim_K K[\boldsymbol{x}] \Big/ \Big\langle f, \frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n} \Big\rangle = \sum_{q \in A_{\mathrm{sing}}} \tau_q(f)\,.$$

This formula can be used to check whether a given hypersurface of $\mathbb{A}^n$, $n \geq 2$, is nonsingular outside the origin:

Example 9.36. Consider the bivariate polynomial

$$f = y^2 - 2x^{28}y - 4x^{21}y^{17} + 4x^{14}y^{33} - 8x^7y^{49} + x^{56} + 20y^{65} + 4x^{49}y^{16}$$

with coefficients in $\mathbb{Q}$. Obviously, $p = (0,0) \in \mathrm{V}\big(f, \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\big) \subset \mathbb{A}^2(\mathbb{C})$. We compute the Tjurina number of f at p:

```
> ring S = 0, (x,y), ds;      // the local ring
> poly f = y2-2x28y-4x21y17+4x14y33-8x7y49+x56+20y65+4x49y16;
> ideal I = f, jacob(f);
> vdim(std(I));               // Tjurina number at the origin
2260
```

Readers who are familiar with Arnold's classification of simple hypersurface singularities (see Arnold, Gusein-Zade, and Varchenko (1985)) may conclude that $\mathrm{V}(f)$ has a singularity of type A_{2260} at the origin.[6] That is, there is an analytic automorphism ϕ of $\mathbb{C}\{x, y\}$ such that $f \circ \phi = y^2 - x^{2261}$.

Next, we verify that $\dim_{\mathbb{Q}} \mathbb{Q}[x, y]/\langle f, \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\rangle = \tau_p(f)$, showing in this way that $\mathrm{V}(f) \subset \mathbb{A}^2(\mathbb{C})$ is singular precisely at the origin:

```
> ring R = 0, (x,y), dp;      // the affine ring Q[x,y]
> ideal I = fetch(S,I);
> vdim(std(I));
2260
```

Are there singular points at infinity? To check this, we homogenize f with respect to a third variable z. As a result, we obtain a polynomial $f^{\mathrm{hom}} \in \mathbb{Q}[x, y, z]$ such that $C = \mathrm{V}(f^{\mathrm{hom}}) \subset \mathbb{P}^2(\mathbb{C})$ is the projective closure of $\mathrm{V}(f)$:

```
> ring SH = 0, (x,y,z), dp;
> poly f = fetch(S,f);
> poly fhom = homog(f,z);
```

[6] Note that 65 is the smallest degree for which the existence of a bivariate polynomial defining an A_{2260}-singularity is known. In fact, this example belongs to a series of "world record" examples due to Gusein-Zade and Nekhoroshev (2000).

Since the point of $\mathbb{P}^2(\mathbb{C})$ with coordinates $x = z = 0$ is obviously not on C, it suffices to find the singular points of C in the affine chart $U = \{x = 1\}$ which does not contain the singular point $(0 : 0 : 1)$. Let g be the dehomogenization of f^{hom} with respect to x:

```
> ring R1 = 0, (y,z), dp;
> map phi = SH,1,y,z;
> poly g = phi(fhom);          // fhom in the affine chart (x=1)
> g;
20y65+y2z63-8y49z9+4y33z18-4y17z27-2yz36+4y16+z9
> ideal J = g, jacob(g);
> vdim(std(J));
120
```

Hence, counted with multiplicity, there are precisely 120 singular points of C in the chart U. As $g \in \langle y, z\rangle^9$, the origin $y = z = 0$ is one of these singular points. Let us compute its multiplicity as a singular point, that is, the Tjurina number of g at the origin:

```
> ring R2 = 0, (y,z), ds;      // the local ring at (1:0:0)
> ideal J = fetch(R1,J);
> vdim(std(J));
120
```

We conclude that $(1:0:0)$ is the only singular point of C at infinity. □

Remark 9.37. (1) The Tjurina number is an important invariant of singularity theory. It is a measure of how complicated the analytic structure of a hypersurface singularity is.

(2) Closely related to the Tjurina number is the Milnor number which is an important topological invariant in the study of critical points of complex functions (see Milnor (1968)). Given a polynomial $f \in \mathbb{C}[x_1, \dots, x_n]$, the **Milnor number** of f at the origin $p \in \mathbb{A}^n$, written $\mu_p(f)$, is defined to be

$$\mu_p(f) := \dim_{\mathbb{C}} \mathbb{C}\{x_1, \dots, x_n\} \Big/ \Big\langle \frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_n} \Big\rangle \, .$$

This definition is extended to an arbitrary point of $\mathbb{A}^n$ by translating the point to the origin.

(3) If $f \in \mathbb{C}[x_1, \dots, x_n]$ is **weighted homogeneous**, that is, if all terms of f have the same $\boldsymbol{w}$-weighted degree for some weight vector $\boldsymbol{w} = (w_1, \dots, w_n) \in \mathbb{Z}^n_{>0}$, then $\tau_p(f) = \mu_p(f)$ due to **Euler's formula**:

$$\boldsymbol{w}\text{-deg}(f) \cdot f = \sum_{i=1}^{n} w_i \cdot x_i \cdot \frac{\partial f}{\partial x_i} \, .$$

In fact, the equality $\mu_p = \tau_p < \infty$ characterizes weighted homogeneous polynomials defining isolated singularities (see Saito (1971) for a precise statement).

(4) The Tjurina and the Milnor number can be defined (and computed with `SINGULAR` using the commands `tjurina` and `milnor` from `sing.lib`) in the more general setting of complete intersection singularities (see Looijenga (1984)). □

In the case of n equations in n variables, multiplicities of solutions are usually referred to as intersection multiplicities. We treat the case $n = 2$, that is, the case of plane curves, demonstrating in an example that, in case $K \subset \mathbb{C}$, intersection multiplicities can be visualized by appropriately perturbing the defining equations.

Remark-Definition 9.38. Let $f, g \in K[x, y] \setminus \{0\}$. The **intersection multiplicity** of f and g at the origin $p \in \mathbb{A}^2$ is defined to be

$$i_p(f, g) = \dim_K K[x, y]_{\langle x,y\rangle}/\langle f, g\rangle \ .$$

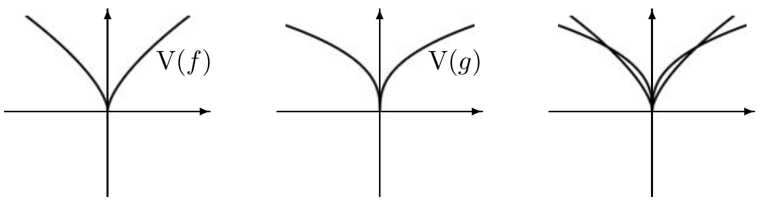
Again, this definition is extended to an arbitrary point $q \in \mathbb{A}^n$ by translating q to the origin.

We have $i_q(f, g) < \infty$ iff f and g have no common factor which vanishes at q. In this case, $i_q(f, g)$ is the multiplicity of q as a solution of $\langle f, g\rangle$. If $i_q(f, g) = 1$, we say that f and g intersect **transversally** at q. □

Geometrically, consider the curves $C = \mathrm{V}(f)$ and $D = \mathrm{V}(g)$ in $\mathbb{A}^2$. If f and g are square-free as polynomials in $\overline{K}[x, y]$, then f and g generate the vanishing ideals of C and D, and we refer to $i_p(f, g)$ also as the **intersection multiplicity** of C and D at p, written $i_p(C, D)$. Similarly, we say that C and D intersect **transversally** at p if this holds for f and g. In this case, p is a smooth point of both C and D, and the two curves have different tangent lines at p.

Example 9.39. We define two affine plane curves which both have a singularity at the origin $p \in \mathbb{A}^2(\mathbb{C})$ and compute their intersection multiplicity at p:

```
> ring R_loc = 0, (x,y), ds;    // local monomial order
> poly f, g = 2x2-y3, 2x2-y5;
> ideal I = f, g;
> vdim(std(I));  // intersection multiplicity at origin
6
> LIB "surf.lib";
> plot(f,"scale_x=0.15; scale_y=0.15;");
> plot(g,"scale_x=0.15; scale_y=0.15;");
> plot(f*g,"scale_x=0.15; scale_y=0.15;");
```



As the rightmost picture indicates, $\mathrm{V}(f)$ and $\mathrm{V}(g)$ intersect at the origin and at further points of $\mathbb{A}^2(\mathbb{C})$. According to the formula (9.3), we may compute the sum of the intersection multiplicities at all points of intersection as follows:

```
> ring R_aff = 0, (x,y), dp;    // global monomial order
> ideal I = imap(R_loc,I);
> vdim(std(I));
10
```

The coordinates of the intersection points (and the intersection multiplicities at these points) can be read from a primary decomposition of `I` in $\mathbb{Q}[x, y]$ (see Remark 9.34):

```
> LIB "primdec.lib";
> primdecGTZ(I);
[1]:
   [1]:
      _[1]=y3
      _[2]=x2
   [2]:
      _[1]=y
      _[2]=x
[2]:
   [1]:
      _[1]=y-1
      _[2]=2x2-1
   [2]:
      _[1]=y-1
      _[2]=2x2-1
[3]:
   [1]:
      _[1]=y+1
      _[2]=2x2+1
   [2]:
      _[1]=y+1
      _[2]=2x2+1
```

Hence, in contrast to the behavior at the origin p, $\mathrm{V}(f)$ and $\mathrm{V}(g)$ intersect transversally (and are smooth) at the other intersection points. Indeed, the primary components defining the intersection points other than p coincide with their radical, so all solutions of $\langle f, g\rangle$ in $\mathbb{A}^2(\mathbb{C}) \setminus \{p\}$ have multiplicity 1. Note that the points corresponding to the third component are not real and, hence, invisible in the picture printed above.

Due to the principle of conservation of numbers (see de Jong and Pfister (2000), Section 6.4), the intersection multiplicity 6 of $\mathrm{V}(f)$ and $\mathrm{V}(g)$ at the origin p can be visualized by slightly perturbing f and g. Indeed, consider two polynomials $F, G \in \mathbb{Q}[x, y, t]$ such that $F(x, y, 0) = f$, $G(x, y, 0) = g$, and set $f_c := F(x, y, c)$, $g_c := G(x, y, c)$ for each $c \in \mathbb{A}^1(\mathbb{C})$. Then the principle implies that there is an ε-neighborhood U of p such that $\sum_{q\in U} i_q(f_c, g_c) = i_p(f, g) = 6$, for each sufficiently small $c \in \mathbb{C}$.[7] Choosing F and G suitably, we get, for each sufficiently small $c \in \mathbb{R} \setminus \{0\}$, six distinct intersection points near p which are all real. For instance, as the following computations show,

$$F = f - t \cdot \frac{y^2}{2}, \quad G = g + t^2 \cdot \left(\frac{5}{4}y^3 - \frac{5t^2}{16}y - \frac{t^3}{16}\right)$$

are just right.

```
> ring Rt_loc = (0,t), (x,y), ds;  // choose coefficient field Q(t)
> poly F = imap(R_loc,f)-(t/2)*y2;
> poly G = imap(R_loc,g)+(5t2/4)*y3-(5t4/16)*y-(t5/16);
> ideal I = F, G;
> vdim(std(I));   // intersection number at (0,0) for general fiber
0
> ring Rt_aff = (0,t), (x,y), dp;
> ideal I = imap(Rt_loc,I);
> vdim(std(I));   // sum of intersection numbers for general fiber
10
```

Thus, for a general choice of $c \in \mathbb{C}$, the curves $\mathrm{V}(f_c)$ and $\mathrm{V}(g_c)$ do not intersect at the origin. Further, counted with multiplicity, there are 10 intersection points in $\mathbb{A}^2(\mathbb{C})$. In fact, all multiplicities are 1, that is, the curves intersect transversally (and are smooth) at ten distinct intersection points. To see this, we apply again Remark 9.34, verifying that the primary ideal defining these points is already radical:

```
> primdecGTZ(I);
[1]:
   [1]:
      _[1]=16*y5+(-20t2-16)*y3+(-8t)*y2+(5t4)*y+(t5)
      _[2]=-2*y3+4*x2+(-t)*y2
   [2]:
      _[1]=16*y5+(-20t2-16)*y3+(-8t)*y2+(5t4)*y+(t5)
      _[2]=-2*y3+4*x2+(-t)*y2
```

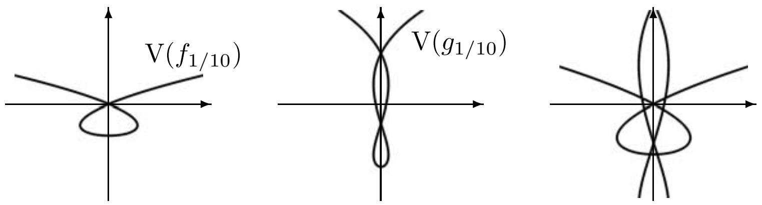
Next, we plot $\mathrm{V}(f_{1/10})$ and $\mathrm{V}(g_{1/10})$ using a high magnification. For a third picture, to be able to distinguish the 6 intersection points near the origin, we once more raise the magnification:

[7] More generally, if $F_1, \dots, F_r \in \mathbb{C}[\boldsymbol{x}, t]$ are such that $\mathbb{C}\{\boldsymbol{x}, t\}/\langle F_1, \dots, F_r\rangle$ is a *flat* $\mathbb{C}\{t\}$-module, then the principle of conservation of numbers implies that there is an ε-neighborhood U of p such that, for each sufficiently small $c \in \mathbb{C}$,

$$\operatorname{mult}\big(p \,\big|\, \langle f_1, \dots, f_r\rangle\big) = \sum_{q\in U} \operatorname{mult}\big(q \,\big|\, \langle f_{1,c}, \dots, f_{r,c}\rangle\big)\,.$$

Here, $f_{i,c} = F_i(\boldsymbol{x}, c) \in \mathbb{C}[\boldsymbol{x}]$, $f_i = f_{i,0}$, for all i. In our situation, the flatness condition is automatically satisfied due to Lecture 5, Theorem 5.12 and Remark 5.13.

```
> setring Rt_loc;
> poly f_def = subst(F,t,1/10);
> poly g_def = subst(G,t,1/10);
> setring R_loc;
> poly f_def = imap(Rt_loc,f_def);
> poly g_def = imap(Rt_loc,g_def);
> plot(f_def,"scale_x=0.001; scale_y=0.015;");
> plot(g_def,"scale_x=0.001; scale_y=0.015;");
> map phi = R_loc, 1/1000*x, 1/10*y;              // zooming map
> plot(10000000*phi(f_def)*phi(g_def),"scale_x=0.5;scale_y=0.075;");
```



The specific families of curves considered above can also be used to visualize the complexity of the singularities of $V(f)$ and $V(g)$ at the origin as measured by the Tjurina and the Milnor number (which coincide in both cases since f and g are weighted homogeneous):

```
> LIB "sing.lib";
> tjurina(f);
2
> milnor(f);
2
> tjurina(g);
4
> milnor(g);
4
```

In a flat family of plane curves over $\mathbb{A}^1(\mathbb{C})$, defined by a polynomial H in $\mathbb{C}[x, y, t]$ such that $h_0 = H(x, y, 0)$ is square-free, the Tjurina number is semi-continuous in the following sense: setting $h_c = H(x, y, c)$, $c \in \mathbb{C}$, there is an ε-neighborhood $U \subset \mathbb{A}^2(\mathbb{C})$ of the origin p such that $\sum_{q \in U} \tau_q(h_c) \leq \tau_p(h_0)$, for each sufficiently small c (see Greuel, Lossen, and Shustin (2006)).

For the families defined by F and G above, this inequality becomes strict. In fact, the singularity of $V(f)$ at the origin, which is a simple cusp (an A_2-singularity in Arnold's classification), is deformed into a node at the origin:[8]

```
> setring Rt_loc;
> tjurina(F);     // Tjurina number at origin for general fiber
1
```

[8] Note that all analytic germs of algebraic sets with Tjurina number 1 (respectively 2) are isomorphic and referred to as **nodes** (respectively **simple cusps**).

```
> setring Rt_aff;
> poly F = imap(Rt_loc,F);
> tjurina(F);     // sum of Tjurina numbers for general fiber
1
```

Further, the A_4-singularity of $V(g)$ splits up into two nodes for $c \neq 0$ close to 0 (none of the nodes is at the origin):

```
> poly G = imap(Rt_loc,G);
> tjurina(G);
2
> setring Rt_loc;
> tjurina(G);
0
```

Observe that, in both cases, the number of nodes appearing on the general fiber is the maximal possible one obtained by deforming a simple cusp, respectively an A_4-singularity (see Slodowy (1980), Section 8.10 for a general discussion of possible degenerations of simple hypersurface singularities).

While the sum of Tjurina numbers decreases if c becomes nonzero, the sum of Milnor numbers for the polynomial functions f_c, g_c remains the same. This follows already from the considerations above since the Milnor number of a polynomial $h \in \mathbb{C}[x,y]$ at a point $q \in \mathbb{A}^2(\mathbb{C})$ is nothing but an intersection multiplicity, namely the intersection multiplicity at q of two **polars** (that is, linear combinations of the partial derivatives) of h.

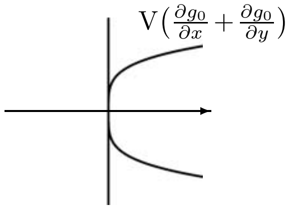
```
> setring Rt_aff;
> milnor(F);        // sum of Milnor numbers for general fiber
2
> milnor(G);
4
```

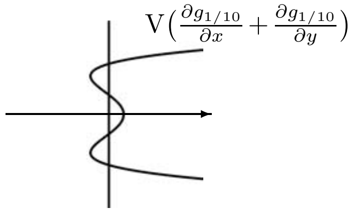
To get illuminative pictures, we have to choose square-free polars. For the family defined by G, we consider

$$H = \frac{\partial G}{\partial x} = 4x\,, \quad H' = \frac{\partial G}{\partial x} + \frac{\partial G}{\partial y} = 4x - 5y^4 + \frac{15}{4}t^2y^2 - \frac{5}{16}t^4$$

and we write $h_c = H(x,y,c)$, $h'_c = H'(x,y,c) \in \mathbb{C}[x,y]$. The intersection point with multiplicity 4 of h_0 and h'_0 at the origin splits up into 4 points of transversal intersection of h_c and h'_c for $c \neq 0$ close to 0 (at each of these four intersection points, the Milnor number of g_c is 1):

```
> setring R_loc;
> poly h0 = diff(g,x);
> poly h0_prime = diff(g,x)+diff(g,y);
> poly hc = diff(g_def,x);
> poly hc_prime = diff(g_def,x)+diff(g_def,y);
> plot(h0*h0_prime,"scale_x=0.15; scale_y=0.15;");
> plot(hc*hc_prime,"scale_x=0.000005; scale_y=0.015;");
```


$V\left(\frac{\partial g_0}{\partial x} + \frac{\partial g_0}{\partial y}\right)$


$V\left(\frac{\partial g_{1/10}}{\partial x} + \frac{\partial g_{1/10}}{\partial y}\right)$

The four intersection points in the right picture (and, thus, the Milnor number 4) can be rediscovered in the real picture of $V(g_{1/10})$ printed earlier as nodes, respectively centers of loops. Similarly for the family defined by F. □

Besides using standard bases, intersection multiplicities of plane curves can also be computed via Hamburger-Noether expansions (see Section 9.6, Remark 9.47 later in this lecture).

9.5 Elimination

In this section, we consider the following problem. Given an ideal $I \subset K[\boldsymbol{x}, \boldsymbol{y}]$ such that $A = V(I) \subset \mathbb{A}^{n+m} = \mathbb{A}^{n+m}(\overline{K})$ contains the origin p of $\mathbb{A}^{n+m}$, compute the image of (the germ of) the linear projection

$$\pi : (A, p) \to (\mathbb{A}^m, o)$$

onto the $\boldsymbol{y}$-components, provided that this image is well-defined.

Remark 9.40. A **continuous map of germs** $(A, p) \to (B, q)$ is obtained by identifying continuous maps between representatives of (A, p) and (B, q) if their restrictions to a suitable neighborhood of p in A coincide. In general, it does not make sense to talk about the image of such a map. As an example, consider the blow-up map

$$\pi : A = V(y - xz) \to \mathbb{A}^2, \quad (x, y, z) \mapsto (y, z),$$

and note that there are arbitrarily small representatives A_1 and A_2 of the germ of A at the origin of $\mathbb{A}^3$ such that for each neighborhood U of the origin of $\mathbb{A}^2$ we have $U \cap \pi(A_1) \neq U \cap \pi(A_2)$. If f is a finite map of germs however, the image is well-defined. Here, **finite** means that there is a continuous representative of f which is closed, and which has finite fibers (see de Jong and Pfister (2000), Section 3.4 for details). □

In the case of algebraic germs, computing the image of a finite linear projection π amounts to computing the **elimination ideal**

$$(I\, K[\boldsymbol{x}, \boldsymbol{y}]_{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}) \cap K[\boldsymbol{y}]_{\langle \boldsymbol{y} \rangle}\,.$$

Comparing the elimination problem in the local ring $K[\boldsymbol{x}, \boldsymbol{y}]_{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}$ to the one in the polynomial ring $K[\boldsymbol{x}, \boldsymbol{y}]$, we detect an extra difficulty. Namely, *no local* *monomial order on* $K[\boldsymbol{x},\boldsymbol{y}]$ *has the elimination property with respect to* $\boldsymbol{x}$. Indeed, if $1 > x_i$, then $\mathrm{L}_>(1+x_i) = 1 \in K[\boldsymbol{y}]$, but $1 + x_i \notin K[\boldsymbol{y}]$. In contrast, any mixed order obtained as the product of a global order on $K[\boldsymbol{x}]$ and a local order on $K[\boldsymbol{y}]$ has the desired elimination property. Recall that such an order implements the ring $K[\boldsymbol{y}]_{\langle \boldsymbol{y}\rangle}[\boldsymbol{x}]$. This suggests to solve the elimination problem in $K[\boldsymbol{x},\boldsymbol{y}]_{\langle \boldsymbol{x},\boldsymbol{y}\rangle}$ via the chain of rings

$$K[\boldsymbol{y}]_{\langle \boldsymbol{y}\rangle} \subset K[\boldsymbol{y}]_{\langle \boldsymbol{y}\rangle}[\boldsymbol{x}] \subset K[\boldsymbol{x},\boldsymbol{y}]_{\langle \boldsymbol{x},\boldsymbol{y}\rangle}\,.$$

In a first step, intersect the given ideal with the subring $K[\boldsymbol{y}]_{\langle \boldsymbol{y}\rangle}[\boldsymbol{x}]$. Then, use a mixed monomial order as above to eliminate the $\boldsymbol{x}$-variables.

For the first step, recall from Remark 9.28 that if $I = \bigcap_{i=1}^s Q_i$ is a primary decomposition, and if $I_0 = \bigcap_{Q_i \subset \langle \boldsymbol{x},\boldsymbol{y}\rangle} Q_i$, then $I\,K[\boldsymbol{x},\boldsymbol{y}]_{\langle \boldsymbol{x},\boldsymbol{y}\rangle} = I_0\,K[\boldsymbol{x},\boldsymbol{y}]_{\langle \boldsymbol{x},\boldsymbol{y}\rangle}$ (geometrically, the irreducible components of A not passing through p do not contribute to defining the germ (A,p)). It easily follows that each set of polynomial generators $\{f_1,\dots,f_r\}$ for I_0 also generates the contracted ideal $I' = (I\,K[\boldsymbol{x},\boldsymbol{y}]_{\langle \boldsymbol{x},\boldsymbol{y}\rangle}) \cap K[\boldsymbol{y}]_{\langle \boldsymbol{y}\rangle}[\boldsymbol{x}]$.

Indeed, for each $g \in I'$ there are polynomials $u_1 \in K[\boldsymbol{y}]$, $u \in K[\boldsymbol{x},\boldsymbol{y}]$ such that $u_1 g \in K[\boldsymbol{x},\boldsymbol{y}]$, $uu_1 g \in I_0$, and $u(\mathbf{0},\mathbf{0}) \cdot u_1(\mathbf{0}) \neq 0$. Since I_0 is an intersection of primary ideals $Q_i \subset \langle \boldsymbol{x},\boldsymbol{y}\rangle$ and since $u^k \notin \langle \boldsymbol{x},\boldsymbol{y}\rangle$ for all $k \in \mathbb{N}$, it follows that $u_1 g \in I_0$. But, this implies that g can be written as a linear combination $g = \sum_{i=1}^r \frac{a_i}{u_1} f_i$, where $a_1,\dots,a_r \in K[\boldsymbol{x},\boldsymbol{y}]$.

Unfortunately, other than computing a primary decomposition of I, we do not know any general method for computing a set of generators for I_0 from a given set of generators for I.

Remark 9.41. Applying the `SINGULAR` command `eliminate` in local rings has to be done with some care since the first step described above is not automatically performed. As an example, consider the ideal $I\,\mathbb{Q}[x,y,z]_{\langle x,y,z\rangle}$, where

$$I = I_0 \cap I_1 = \langle x-z, y-z^2\rangle \cap \langle x-z+1, y-z^2+1\rangle \subset \mathbb{Q}[x,y,z]\,.$$

We begin by defining I:

```
> ring R = 0, (x,y,z), dp;
> ideal I0 = x-z, y-z^2;       // vanishing at the origin
> ideal I1 = x-z+1, y-z^2+1;   // not vanishing at the origin
> ideal I = intersect(I0,I1);
> I = std(I); I;
_[1]=z2+x-y-z
_[2]=x2-2xz+y
```

Note that I contains the polynomial $z^2 + x - y - z$ which is monic in z. This implies that the projection $\mathrm{V}(I) \to \mathbb{A}^2(\mathbb{C})$, $(x,y,z) \mapsto (x,y)$, is finite (see Theorem 6.21).

Next, we turn to the local ring:

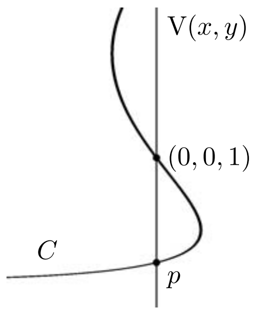
```
> ring Rloc = 0, (x,y,z), ds;
> ideal Iloc = fetch(R,I);
> Iloc = std(Iloc); Iloc;
_[1]=x-y-z+z2
_[2]=y+x2-2xz
```

In $\mathbb{Q}[x, y, z]$, the polynomials originating from the standard basis computation in the local ring generate the ideal I which is a proper subset of I_0. As a consequence, entering `eliminate(Iloc,z);` results in an ideal which does not define the desired image:

```
> ideal J = eliminate(Iloc,z);
> J;
J[1]=2xy-y2-2x3+2x2y-x4
> factorize(J[1],1);
_[1]=-2x+y-x2
_[2]=-y+x2
> ideal I0loc = fetch(R,I0);
> eliminate(I0loc,z);
_[1]=y-x2
```
□

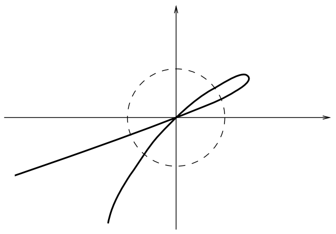
In the case of analytic germs, we do not know a general method for computing the image of π. In fact, computing polynomial generators for the elimination ideal $(I\,K[\boldsymbol{x}, \boldsymbol{y}]_{\langle \boldsymbol{x},\boldsymbol{y}\rangle}) \cap K[\boldsymbol{y}]_{\langle \boldsymbol{y}\rangle}$ as above may not be sufficient. Namely, if $\widetilde{A} \subset \mathbb{A}^m$ is the algebraic set defined by these generators, the image of π as an analytic germ is contained in the analytic germ $(\widetilde{A}, o)$, but the containment may well be strict:

Example 9.42. Consider the smooth irreducible space curve C parametrized by $t \mapsto (t-t^3, t-t^2, t)$.



The following `SINGULAR` session computes defining equations for the analytic germ of C at the origin $p \in \mathbb{A}^3(\mathbb{C})$, and it eliminates z from these equations:

```
> ring R = 0, (t,x,y,z), ds;
> ideal I = x-t+t3, y-t+t2, z-t;
> ideal C = eliminate(I,t); C;
C[1]=x-z+z3
C[2]=y-z+z2
> eliminate(C,z);
_[1]=x2-3xy+2y2+y3
> plot(x2-3xy+2y2+y3,"scale_x=0.05;scale_y=0.05;");
```



The analytic germ defined by $x^2 - 3xy + 2y^2 + y^3$ at the origin of $\mathbb{A}^2(\mathbb{C})$ consists of two smooth branches intersecting transversally. As C is smooth at p, the image of (C, p) under the linear projection onto the xy-plane can only be *one* of these branches. □

Note, however, that if $\overline{A}$ denotes the Zariski closure of A in $\mathbb{P}^n \times \mathbb{A}^m$ and if the fiber of the projection $\overline{A} \to \mathbb{A}^m$ over the origin o consists of the origin p only, then the image of π coincides with $(\widetilde{A}, o)$.

9.6 Hamburger-Noether Expansion

If it comes to factorizing a polynomial in $K[[\boldsymbol{x}]]$, the only case which can be tackled algorithmically is the case of two variables (geometrically, the case of germs of plane curves).

Definition 9.43. Let $0 \neq f \in K[[x, y]]$, and let $f = c \cdot f_1^{m_1} \cdot \ldots \cdot f_s^{m_s}$ be the **absolute factorization** of f, that is, the factorization of f as an element of $\overline{K}[[x, y]]$. Then we call $f_1, \ldots, f_s$ the **branches** of f. Each such branch is a power series with coefficients in some finite extension of K. □

The name branch comes from geometry. If $K \subset \mathbb{C}$, and if $0 \neq f \in K\{x, y\}$ is a convergent power series, each branch of f is a convergent power series, too. It, thus, defines the germ of an analytic set at the origin $p \in \mathbb{A}^2(\mathbb{C})$. This germ is a **branch** (an irreducible component) of the analytic germ of $\mathrm{V}(f)$ at p.

A classical approach to computing branches goes back to Newton. His constructive method yields the following result (see Brieskorn and Knörrer (1986) or Greuel, Lossen, and Shustin (2006) for details):

Theorem 9.44. *Let $f \in \mathbb{C}\{x,y\} \setminus \{0\}$ be a convergent power series which is irreducible and y-general of order $m > 0$. There exists a power series $y(t) \in \mathbb{C}\{t\}$ such that $y(0) = 0$, and such that $\varphi : t \mapsto \big(t^m, y(t)\big)$ defines a parametrization of the analytic germ of* $\mathrm{V}(f)$ *at the origin of* $\mathbb{A}^2(\mathbb{C})$.

We refer to the fractional power series $y(x^{1/m}) \in \mathbb{C}\{x^{1/m}\}$ as a **Puiseux expansion** for f.

Remark 9.45. More generally, let K be any field of characteristic zero. If $f \in K[[x,y]] \setminus \{0\}$ is square-free, Newton's method yields for each branch f_i of f a parametrization of type $t \mapsto \big(t^{m_i}, y_i(t)\big)$ or $t \mapsto \big(x_i(t), t^{m_i}\big)$. Here, no a priori knowledge on the decomposition of f into branches is required. In particular, Newton's method allows us to compute the number of branches and, via elimination, the power series expansion of each branch (up to any given degree). We will return to this in Remark 9.49. $\square$

In positive characteristic, Puiseux expansions do not necessarily exist. Instead, there is the concept of Hamburger-Noether expansions which works in any characteristic.

Definition 9.46. Let K be any field, and let $f \in K[[x,y]] \setminus \{0\}$ be **absolutely irreducible** (that is, f is irreducible in $\overline{K}[[x,y]]$). Suppose that x is a **transversal parameter** for f, that is, there is a lowest degree term of f which is contained in $K[y]$. A **Hamburger-Noether expansion** for f consists of a system of equations in the variables $z_{-1}, z_0, \ldots, z_\ell$ of type

$$\begin{aligned}
z_{-1} &= a_{0,1}z_0 + a_{0,2}z_0^2 + \ldots + a_{0,h_0}z_0^{h_0} + z_0^{h_0}z_1\\
z_0 &= \phantom{a_{0,1}z_0 +{}} a_{1,2}z_1^2 + \ldots + a_{1,h_1}z_1^{h_1} + z_1^{h_1}z_2\\
&\;\;\vdots \qquad\qquad \vdots\\
z_{i-1} &= \phantom{a_{0,1}z_0 +{}} a_{i,2}z_i^2 + \ldots + a_{i,h_i}z_i^{h_i} + z_i^{h_i}z_{i+1}\\
&\;\;\vdots \qquad\qquad \vdots\\
z_{\ell-2} &= \phantom{a_{0,1}z_0 +{}} a_{\ell-1,2}z_{\ell-1}^2 + \ldots + a_{\ell-1,h_{\ell-1}}z_{\ell-1}^{h_{\ell-1}} + z_{\ell-1}^{h_{\ell-1}}z_\ell\\
z_{\ell-1} &= \phantom{a_{0,1}z_0 +{}} \sum_{j\geq 2} a_{\ell,j}z_\ell^j\,,
\end{aligned}$$

where ℓ is a nonnegative integer, the coefficients $a_{i,j}$ are elements of K, the h_i are positive integers, and

$$f\big(z_0(z_\ell), z_{-1}(z_\ell)\big) = 0 \in K[[z_\ell]]\,.$$

Here, the power series $z_0(z_\ell), z_{-1}(z_\ell) \in K[[z_\ell]]$ are obtained as follows. Let $z_{\ell-1}(z_\ell) \in K[[z_\ell]]$ denote the power series on the right-hand side of the last equation in the Hamburger-Noether expansion. Then, for $j = \ell-1, \ldots, 0$, define $z_{j-1}(z_\ell) \in K[[z_\ell]]$ by substituting $z_j(z_\ell)$ for z_j and $z_{j+1}(z_\ell)$ for z_{j+1} on the right-hand side of the equation for z_{j-1}.

If the last equation of a system as above is replaced by an implicit equation of type $g(z_\ell, z_{\ell-1}) = 0$, where $g \in K[x,y]$ is a polynomial satisfying $\frac{\partial g}{\partial y}(0,0) \neq 0$, then the system is called a **symbolic Hamburger-Noether expression** for f. $\square$

In contrast to a Hamburger-Noether expansion, the data in a symbolic Hamburger-Noether expression are all polynomial. Given a symbolic Hamburger-Noether expression, a corresponding expansion can be computed up to any given degree via the implicit function theorem.

If $K \subset \mathbb{C}$, and if f is convergent, then $z_0(z_\ell)$ and $z_{-1}(z_\ell)$ are convergent, too. In this case, the map $t \mapsto \big(z_0(t), z_{-1}(t)\big)$ defines a parametrization of the analytic germ of $\mathrm{V}(f)$ at the origin of $\mathbb{A}^2(\mathbb{C})$.

Remark 9.47. (1) Campillo (1980) showed that every nonzero polynomial $f \in K[x,y]$ which is irreducible as a power series in $\overline{K}[[x,y]]$ admits a symbolic Hamburger-Noether expression (if x is not a transversal parameter for f, the roles of $z_0(z_\ell)$ and $z_{-1}(z_\ell)$ have to be interchanged). The proof is constructive and yields an algorithm for computing the expression. More generally, there is an algorithm due to Rybowicz (1990) taking as input a polynomial $0 \neq f \in K[x,y]$ which is square-free in $\overline{K}[[x,y]]$, and computing for each branch of f a symbolic Hamburger-Noether expression with coefficients in a finite extension of K. As for Puiseux expansions, no a priori information on the decomposition of f into branches is required. In particular, a finite extension of K over which the absolute factorization of f as a formal power series occurs is (successively) specified in the process. A variant of the algorithm of Rybowicz is implemented in the `SINGULAR` library `hnoether.lib` and is accessible via the `hnexpansion` command.

(2) Given nonzero power series $f, g \in K[[x,y]]$, the **intersection multiplicity** of f and g at the origin $p \in \mathbb{A}^2$ is defined to be

$$i_p(f,g) = \dim_K K[[x,y]]/\langle f, g\rangle \,.$$

One can show that if f is as in Definition 9.46, and if $z_0(z_\ell), z_{-1}(z_\ell) \in K[[z_\ell]]$ are the power series obtained from a Hamburger-Noether expansion for f, then

$$i_p(f,g) = \mathrm{ord}_{z_\ell}\, g\big(z_0(z_\ell), z_{-1}(z_\ell)\big) \,.$$

Here, if $h \in K[[z_\ell]] \setminus \{0\}$, we write $\mathrm{ord}_{z_\ell}(h)$ for the lowest degree of a term of h. Moreover, $\mathrm{ord}_{z_\ell}(0) = \infty$.

(3) If f, g are polynomials in $K[x,y]$, then, by Proposition 9.4, the intersection multiplicity at p defined in (2) coincides with that defined earlier. Note that if f is square-free over $\overline{K}$, and if $f_1, \dots, f_s \in \overline{K}[[x,y]]$ are the branches of f, then

$$i_p(f,g) = \sum_{i=1}^{s} \dim_{\overline{K}} \overline{K}[[x,y]]/\langle f_i, g\rangle \,.$$

This sum can be computed over any finite extension of K over which the branches are defined.

For a proof of (2) and (3), we refer to Greuel, Lossen, and Shustin (2006). For (2), see also de Jong and Pfister (2000), Lemma 5.1.5. □

Example 9.48. We continue our SINGULAR session from Example 9.27, now computing symbolic Hamburger-Noether expressions for the branches of f. We use the hnexpansion command.

```
> LIB "hnoether.lib";
> list L = hnexpansion(f);
```

According to the information printed by SINGULAR at this point, hnexpansion created a ring in which the symbolic Hamburger-Noether expressions have been computed. Making this ring the active ring, the (truncated) Hamburger-Noether expansions can be recovered from a list named hne:

```
> def HNring = L[1];
> show(HNring);
// ring: (0,a),(x,y),(ls(2),C);
// minpoly = (a2+1)
// objects belonging to this ring:
// f                      [0]  poly
// hne                    [0]  list, size: 6
```

We see that the branches are defined over the finite extension $\mathbb{Q}(i)$ of $\mathbb{Q}$. The size 6 of the list hne is the number of branches. Having made HNring the active ring, we may display the computed (truncated) Hamburger-Noether expansions of the individual branches:

```
> setring HNring;
> displayHNE(hne);
// Hamburger-Noether development of branch nr.1:
  y = x+1/2*x^2 + ..... (terms of degree >=3)

// Hamburger-Noether development of branch nr.2:
  y = -x-1/2*x^2 + ..... (terms of degree >=3)

// Hamburger-Noether development of branch nr.3:
  y = z(1)*x
  x = z(1)^2+z(1)^2*z(2)
  z(1) = 1/4*z(2)^2-1/2*z(2)^3 + ..... (terms of degree >=4)

// Hamburger-Noether development of branch nr.4:
  x = z(1)*y
  y = -z(1)^2

// Hamburger-Noether development of branch nr.5:
  x = (a)*y^2

// Hamburger-Noether development of branch nr.6:
  x = (-a)*y^2
```

For the branches numbered $1, 2, 5, 6$, we can directly read the corresponding factors of f in $\mathbb{C}[[x, y]]$:

- Factor #1: $y - x - \frac{1}{2}x^2 + g(x)$, with $g \in \langle x\rangle^3 \subset \mathbb{Q}[[x]]$.
- Factor #2: $y + x + \frac{1}{2}x^2 + h(x)$, with $h \in \langle x\rangle^3 \subset \mathbb{Q}[[x]]$.
- Factor #5: $x - iy^2$.
- Factor #6: $x + iy^2$.

For the 4th branch, we get the equation $x = \sqrt{-y} \cdot y$, hence:

- Factor #4: $x^2 + y^3$.

For the 3rd branch, we compute a parametrization using the `param` command from `hnoether.lib`:

```
> ring R_param = (0,a), (x,y,t), ls;
> minpoly = a2+1;
> list hne = imap(HNring,hne);
> list P = param(hne[3],1);
// ** Warning: result is exact up to order 5 in x and 7 in y !
> P[1];
_[1]=1/16*t4-3/16*t5+1/4*t7
_[2]=1/64*t6-5/64*t7+3/32*t8+1/16*t9-1/8*t10
```

We see that the computed parametrization is exact up to degree 5 in the first component and up to degree 7 in the second one. Exactness for higher degrees is obtained by computing further terms on the right-hand side of the last row of the Hamburger-Noether expansion `z(1) = 1/4*z(2)^2-1/2*z(2)^3 + ...`. For instance:

```
> hne[3] = extdevelop(hne[3],5);
> P = param(hne[3],1);
// ** Warning: result is exact up to order 7 in x and 9 in y !
> P[1];
_[1]=1/16*t4-3/16*t5+47/128*t6-19/32*t7+ [...]
_[2]=1/64*t6-5/64*t7+237/1024*t8-271/512*t9+ [...]
```

We read the parametrization of the third branch:

$$t \longmapsto \Big(\tfrac{1}{16}t^4 - \tfrac{3}{16}t^5 + \tfrac{47}{128}t^6 - \tfrac{19}{32}t^7 + \dots,\\ \tfrac{1}{64}t^6 - \tfrac{5}{64}t^7 + \tfrac{237}{1024}t^8 - \tfrac{271}{512}t^9 + \dots\Big) .$$

More information on the branches is obtained via the `displayInvariants` command. In particular, the intersection multiplicities are printed (for a discussion of the other invariants shown below, we refer to the literature listed in Remark 9.50 at the end of this lecture):

```
> displayInvariants(hne);
[...]
--- invariants of branch number 3 : ---
```

```
characteristic exponents  : 4,6,7
generators of semigroup   : 4,6,13
Puiseux pairs             : (3,2)(7,2)
degree of the conductor   : 16
delta invariant           : 8
sequence of multiplicities: 4,2,2,1,1
[...]
 ------------- intersection multiplicities : -------------

branch |    6     5     4     3     2
-------+----------------------------
    1  |    1,    1,    2,    4,    1
    2  |    1,    1,    2,    4
    3  |    4,    4,    8
    4  |    3,    3
    5  |    2
[...]
```

□

Remark 9.49. As indicated by the example, Hamburger-Noether expressions are extremely useful in that they carry plenty of geometric information on the branches. Typically, however, there is no way of computing the branches themselves. Here, as already pointed out, Puiseux expansions are better behaved. In the example above, a Puiseux expansion for the third branch is $x^{3/2}+x^{7/4}$, leading to the local parametrization $t \mapsto (t^4, t^6+t^7)$. From this (polynomial) parametrization, we obtain the branch via elimination as explained in Section 9.5:

```
> ring R = 0, (x,y,t), ds;
> ideal J = x-t4, y-t6-t7;
> eliminate(J,t);
_[1]=y4-2x3y2+x6-4x5y-x7
```

The reason for the different behavior of Puiseux and Hamburger-Noether expansions is that in the elimination problem resulting from a parametrization of type $t \mapsto \big(t^m, y(t)\big)$ (as obtained from a truncated Puiseux expansion), the origin is the unique point of

$$\mathrm{V}\big(x-t^m, y-y(t)\big) \subset \mathbb{A}^3(\mathbb{C})$$

which projects to the origin $(0,0)$ of the xy-plane, while for the elimination problem resulting from a truncated Hamburger-Noether expansion, there are usually further points projecting to $(0,0)$. □

Remark 9.50 (Further Reading). For more details and proofs of the results presented in this lecture, see Greuel and Pfister (2002) and Decker and Schreyer (2006). To learn more on analytic germs and on singularities, we refer to Brieskorn and Knörrer (1986), de Jong and Pfister (2000), and Greuel, Lossen, and Shustin (2006).

Practical Session V

Exercise 5.1. (a) Write a `SINGULAR` procedure `min_generating_set`

- which takes as input matrices of primary and secondary invariants of a finite group as provided by the `SINGULAR` command `invariant_ring` from `finvar.lib`, and
- which computes a **fundamental system of invariants**, that is, a minimal generating set for the ring of invariants under consideration.

(b) In Example 8.9 of Lecture 8, compute a fundamental system of invariants. Conclude that, as stated in the lecture, Noether's degree bound does not hold in this case.

Exercise 5.2. (a) Write a `SINGULAR` procedure which takes as input a polynomial f in the active ring, and which returns 1 if f is a unit in the active ring (and 0 otherwise).

(b) Write a `SINGULAR` procedure which takes as input a polynomial f in the active ring and an integer d, and which returns the power series expansion of the inverse of f up to terms of degree d if f is a unit in the active ring (and 0 otherwise).

Exercise 5.3. Consider the curve $C \subset \mathbb{P}^2(\mathbb{C})$ with defining equation

$$\left((x^4+y^4+z^4)^4 - x^9y^5z^2\right) \cdot (x^4+y^4+z^4) \in \mathbb{Q}[x,y,z]\,.$$

For each singular point p of C, make `SINGULAR` compute the number of branches of C at p and the pairwise intersection multiplicities of the branches. Which of the branches are smooth?

Exercise 5.4. Consider the projective closure $C \subset \mathbb{P}^2(\mathbb{C})$ of the affine plane curve with defining equation

$$3y^3 - 3xy^2 - 2xy^3 + x^2y^3 + x^3 = 0\,.$$

(a) Determine the singular locus of C and show that each singular point is either an ordinary multiple point or a simple cusp of C.[9]

(b) Compute the **adjoint linear system** $\mathcal{L}_3$ of C of degree 3. By definition, this system is formed by all homogeneous polynomials $f \in \mathbb{C}[x, y, z]$ of degree 3 satisfying

$$f \in \begin{cases} \mathrm{I}(\{p\})^{m-1} & \text{if } p \text{ is an ordinary } m\text{-multiple point of } C\,, \\ \mathrm{I}(\{p\}) & \text{if } p \text{ is a simple cusp of } C\,. \end{cases}$$

Choose two polynomials f_1, f_2 of $\mathcal{L}_3$ at random and compute the set B_1 (respectively B_2) of intersection points of $\mathrm{V}(f_1)$ (respectively $\mathrm{V}(f_2)$) with C outside the singular locus of C.

(c) Compute the linear system $\mathcal{L}'$ formed by those polynomials in the adjoint linear system of C of degree 4 which vanish along $B_1 \cup B_2$. Choose an element f' of $\mathcal{L}'$ at random and compute the set B_3 of intersection points of $\mathrm{V}(f')$ with C outside the base locus of $\mathcal{L}'$.

(d) Compute the subsystem $\mathcal{L}'' \subset \mathcal{L}_3$ formed by the polynomials which vanish along B_3. Observe that $\mathcal{L}''$ is a pencil, that is, it has projective dimension 1. Choose two elements $f_1'', f_2'' \in \mathcal{L}''$ which span $\mathcal{L}''$.

(e) By construction, the curve $\mathrm{V}(f_1''+af_2'')$ intersects C at 14 fixed points plus one point that moves with $a \in \mathbb{C}$. Use this to compute a **rational parametrization** of (the affine part of) C with rational coefficients. That is, compute $g_i, h_i \in \mathbb{Q}[t]$ such that the image of the map

$$\varphi : \mathbb{A}^1(\mathbb{C}) \longrightarrow \mathbb{A}^2(\mathbb{C})\,, \quad t \longmapsto \left(\frac{g_1(t)}{h_1(t)}, \frac{g_2(t)}{h_2(t)} \right)\,,$$

is dense in C. Use elimination to show that your result is indeed a rational parametrization of C.

[9] Here, a point p is called an **ordinary m-multiple point** of C if the analytic germ (C, p) consists of m smooth branches, intersecting transversally at p. If u, v are local coordinates at p, if $f \in \mathbb{C}\{u, v\}$ is a local equation for C at p, and if $\mathrm{jet}_k(f)$ denotes the **k-jet** of f (that is, the sum of all terms of f up to degree k), then p is an ordinary m-multiple point of C iff $\mathrm{jet}_{m-1}(f) = 0$ and $\mathrm{jet}_m(f)$ decomposes into m pairwise different linear factors. Note that ordinary 2-multiple points are just nodes.

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

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

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 $N = \bigoplus_{i\in\mathbb{Z}} \operatorname{Hom}_K(N_{-i}, K)$ 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 $l \leq h$ 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 **R** and **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

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

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 $\boldsymbol{M}$**, 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 $\boldsymbol{\mathcal{F}}$**.

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 $V(x_0 + x_1 + x_2, x_3, x_4, x_5)$:

```
> 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 $>_{\texttt{dp}}$ 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"

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

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 $V(J) \subset \mathbb{A}^2 \times \mathbb{A}^{10}$ 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

$$\begin{tikzcd} 0 & M \arrow[l] & F_0 = \bigoplus_{i=1}^{s} R(-v_i) \arrow[l] & F_1 \arrow[l, "\varphi"'] \end{tikzcd}$$

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 $e_1,\dots,e_s$ denote the canonical basis vectors of F_0. Then

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

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 $\operatorname{Hom}_R(M,N)$ is isomorphic to $\operatorname{Ext}^0_R(M,N)$, 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 $\operatorname{Ext}^i_R(M,N)$ 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:

$$\begin{tikzcd}
\operatorname{Hom}_R(F_{i+1},N) & \operatorname{Hom}_R(F_i,N) \arrow[l] & \operatorname{Hom}_R(F_{i-1},N) \arrow[l] \\
\operatorname{Hom}_R(F_{i+1},G_0) \arrow[u] & \operatorname{Hom}_R(F_i,G_0) \arrow[l, "\mathtt{a2}"'] \arrow[u] & \operatorname{Hom}_R(F_{i-1},G_0) \arrow[l, "\mathtt{a1}"'] \arrow[u] \\
\operatorname{Hom}_R(F_{i+1},G_1) \arrow[u, "\mathtt{b2}"] & \operatorname{Hom}_R(F_i,G_1) \,. \arrow[l] \arrow[u, "\mathtt{b1}"] &
\end{tikzcd}$$

In this diagram, the relevant maps for constructing $\operatorname{Ext}^i_R(M,N)$ 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:

$$\begin{tikzcd}
0 & S \arrow[l] \arrow[d, equal] & \mathrm{Syz}_3(K(2)) \oplus \mathrm{Syz}_2(K(1))^2 \arrow[l, "\gamma"'] \arrow[d] & N \arrow[l] \arrow[d] & 0 \arrow[l] \\
0 & S \arrow[l] & S^{20} \arrow[l] \arrow[d] & S^{19} \arrow[l, "\widetilde{\gamma}"'] \arrow[d, "\delta"] & 0 \arrow[l] \\
 & & 7S(1) \arrow[r, equal] & 7S(1) &
\end{tikzcd}$$

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:

$$\begin{tikzcd}
0 & M \arrow[l] \arrow[d, "\alpha"'] & S(-1)^{15} \arrow[l] \arrow[d, "\alpha_0"'] & S(-2)^3 \arrow[l, "\varphi"'] \arrow[d, "\alpha_1"'] \\
0 & N \arrow[l] & S(-1)^{25} \arrow[l] & S(-2)^{15} \oplus S(-3)^2 \arrow[l, "\psi"']
\end{tikzcd}$$

```
> 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 $\text{Res}_{2,2,2}$ (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 $\text{Res}_{2,2,2}$ 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 $F = F_a$:

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

```
> 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 $V(F) \subset \mathbb{P}^2$ 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 $f_1, \dots, f_s \in K[\boldsymbol{x}]$ be homogeneous polynomials which are sorted such that $\deg(f_1) \geq \dots \geq \deg(f_s) \geq 1$. Write $L_i = \{1, \dots, s\} \setminus \{i\}$. Suppose that for some j, we have $K[f_\nu \mid \nu \in L_i] \subsetneq K[f_1, \dots, f_s]$, for all $i < j$. Then $K[f_\nu \mid \nu \in L_j] = K[f_1, \dots, f_s]$ iff the ideal $J := \langle y_1 - f_1, \dots, y_s - f_s \rangle \subset K[\boldsymbol{x}, \boldsymbol{y}]$ contains an element of type $y_j - g$, where $g \in K[y_{j+1}, \dots, y_s]$.

```
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

$$(u_0 - u_0u_1)^{-1} = \frac{1}{u_0} \cdot \left(1 + \sum_{k=1}^{\infty} u_1^k\right) \in K[[\boldsymbol{x}]]\,.$$

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 $V(f_1)$ and $V(f_2)$ 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);
```

$\square$

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