Advanced School and Workshop on Singularities in Geometry and Topology

(15 August - 3 September 2005)

Computational Aspects of Singularities

Anne Frühbis-krüger
Universität Kaiserlsautern
Fachbereich Mathematik
Kaiserslautern, Germany

Computational Aspects of Singularities
Notes of a Set of 2 Talks given at the Summer School on Singularities in Geometry and Topology at the ICTP 2005

Anne Frühbis–Krüger

July 22, 2005

Contents

Preface

This is a preliminary version of lecture notes for a set of two talks entitled *Computational Aspects of Singularities* at the *School on Singularities in Geometry and Topology*. The aim of the talks is to first introduce the participants to the use of computational methods for studying singularities and then proceed step by step from the simple application of predefined computational tools to more complex applications. Using the algorithmic resolution of singularities as an example, we further show, how a rather complex computational task can be tackled by decomposing it into several smaller tasks.

As this set of talks is embedded into a school on singularities, we assume that the participants are familiar with the (singularity theory) background of the treated computational tasks or that they become familiar with it during the 3 week period of the school. Hence, we simply put each task into context by recalling the definitions and some properties from the algebraic point of view before outlining the computational approach to it and discussing a practical example (computed in the CA-system SINGULAR).

1 Studying the Singular Locus

In this first section, we show how to use some tools from computational commutative algebra to study the singular locus of a variety. For a more detailed discussion of these techniques and for a description of the underlying algorithms see [?].

1.1 The Jacobian criterion

Given a affine variety $V(I) \subset K^n$ over a (perfect) field K, $I = \langle f_1, \ldots, f_m \rangle \subset K[x_1, \ldots, x_n]$, our first task is to to determine its singular locus by means of the Jacobian criterion.

Recall that the set

$$Sing(A) := \{ P \in Spec(A) \mid A_p \text{ is not regular} \}$$

is called the singular locus of A and that the Jacobian criterion can be stated as follows:

Lemma 1 *Let K be a perfect field, let $A = K[x_1, \ldots, x_n]/\langle f_1, \ldots, f_m \rangle$ be equidimensional and let $J \subset A$ be the ideal generated by the $(n - dim(A))$-minors of the Jacobian matrix $(\frac{\partial f_i}{\partial x_j})$. Then*

$$Sing(A) = V(J).$$

2

Example 1 *(irreducible curve)*
In this first example, we consider the curve specified by the parametrization

$$
\begin{array}{ccc}
\mathbb{A}^1_{\mathbb{C}} & \longrightarrow & \mathbb{A}^3_{\mathbb{C}} \\
t & \longmapsto & (t^3, t^4, t^5).
\end{array}
$$

We shall first compute the ideal of the curve and then apply the Jacobian criterion. At this point, it is important to observe that in general all calculations in a computer algebra system are performed over the rationals or over suitable field extensions thereof, but not over the real or complex numbers. This does not change the calculations, but has to be kept in mind when discussing the computational result.

We now determine the ideal of the curve by specifying the parametrization and then eliminating the parameter t:

```
                               // ring of char 0 containg
> ring r=0,(t,x,y,z),dp;        // variables x,y,z and t
> ideal Ip=x-t^3,y-t^4,z-t^5;   // input of the parametrization

> ideal Ie=eliminate(I,t);      // compute ideal of the curve:
> Ie;                           // elimination provides equations
_[1]=y2-xz                      // of the projection
_[2]=x2y-z2
_[3]=x3-yz
> ring r2=0,(x,y,z),dp;         // we do not need t any more

                               // move the ideal of the
> ideal I=imap(r,Ie);           // curve to this ring
```

As this is obviously an irreducible curve, we do not need to worry about the equidimensionality condition in the Jacobian criterion. We can simply proceed by computing the dimension of $K[x,y,z]/I$ (of which we know that it is 1), the Jacobian matrix and the ideal of minors of the appropriate size:

```
                               // dimension of K[x,y,z]/I;
                               // 'std' (Groebner basis) required
> int dimA=dim(std(I));         // for using command 'dim'
> dimA;                         // we already know that it is 1
1

> matrix Jac=jacob(I);          // determine Jacobian matrix
> print(Jac);                   // show the matrix
-z, 2y,-x,
2xy,x2,-2z,
3x2,-z,-y
```

3

```
> ideal J=minor(Jac,3-dimA);    // determine the minors
> ideal sL=J+I;                 // ideal of singular locus
> sL;
sL[1]=-x2y-2z2
sL[2]=-2xy2+6x2z
sL[3]=-2y2-xz
sL[4]=3x3+yz
sL[5]=3x4+2xyz
sL[6]=6x2y-z2
sL[7]=x3-4yz
sL[8]=2x2y+2z2
sL[9]=4xy2+x2z
sL[10]=y2-xz
sL[11]=x2y-z2
sL[12]=x3-yz
```

12 generators for the ideal of the singular locus seems to be quite a lot. Indeed, ideals generated by minors of matrices tend to have a high number of redundant generators and we can try to find a smaller set of generators by applying appropriate commands such as mstd. *But in our particular case, we are only interested in the set of singular points and we can hence even pass to the radical of the ideal by the Hilbert Nullstellensatz:*

```
                                // the radical is contained in a
> LIB "primdec.lib";            // Singular library which needs to
> radical(sL);                  // be loaded before using it
_[1]=z
_[2]=y
_[3]=x
```

Hence we see that the only singular point of this curve is the origin.

Example 2 *In the second example, we consider a variety consisting of 3 smooth hyperplanes. To this end, we first define each of the hyperplanes separately and then form their union by intersecting the corresponding ideals. Afterwards, we apply the predefined procedure* slocus *to determine the singular locus and subsequently study it further.*

```
> ring r=0,(x,y,z,w),dp;        // polynomial ring: char. 0, 4 var.
> ideal I1=x,w;                 // the y-z plane
> ideal I2=y,z;                 // the x-w plane
> ideal I3=z-x^2-y^2,w;         // another smooth surface
> ideal Itemp=intersect(I1,I2); // union of y-z and x-w planes
> Itemp;
Itemp[1]=yw
Itemp[2]=zw
Itemp[3]=xy
```

```
Itemp[4]=xz
> ideal I=intersect(Itemp,I3);    // union of all three surfaces
> I;
I[1]=zw
I[2]=yw
I[3]=-x2zw-y2zw+z2w
I[4]=-x2yw-y3w+yzw
I[5]=-x3z-xy2z+xz2
I[6]=-x3y-xy3+xyz
I[7]=-y2w+zw
```

In this case, we expect the singular locus to be the locus where the surfaces meet, since each of the surfaces is smooth. The surfaces $V(I1)$ and $V(I2)$ respectively $V(I2)$ and $V(I3)$ meet in the point $V(\langle x, y, z, w \rangle)$, whereas the intersection locus of $V(I1)$ and $V(I3)$ is the curve $V(\langle x, w, z - y^2 \rangle)$.

```
> LIB "sing.lib";               // slocus is in library 'sing.lib'
> ideal sL=slocus(I);          // compute singular locus
> size(sL);
91
```

To see both components of the singular locus, we cannot restrict our consid-erations to the radical or the minimal associated primes in this case, because one component, the point, is contained in the other component. Here we need to consider a primary decomposition[1] of the ideal sL, to find both components:

```
> LIB "primdec.lib";          // library for primary decomposition
> minAssGTZ(sL);              // minimal associated prime ideals
[1]:                          // just one minimal prime
   _[1]=y2-z
   _[2]=w
   _[3]=x
> primdecGTZ(sL);             // complete primary decomposition
[1]:                          // first primary component
   [1]:                       // primary ideal
      _[1]=w
      _[2]=y2-z
      _[3]=x
   [2]:                       // corresponding prime
      _[1]=w
      _[2]=y2-z
      _[3]=x
[2]:                          // 2nd primary component
   [1]:                       // primary ideal
```

---

[1]In the primary decomposition commands implemented in SINGULAR, the list containing the result is not ordered. Therefore permutations of the list entries occur quite often.

5

```
   _[1]=w2
   _[2]=zw
   _[3]=z2
   _[4]=yw
   _[5]=y3z
   _[6]=y4-y2z
   _[7]=xyz
   _[8]=xy2
   _[9]=x3w
   _[10]=x3z
   _[11]=x3y
   _[12]=x6
[2]:                        // corresponding prime
   _[1]=w
   _[2]=z
   _[3]=y
   _[4]=x
```

1.2 The non-equidimensional case

The two previous examples were constructed in a suitable way to make sure that they are equidimensional. But in general this is not known a priori. Hence the variety needs to be decomposed first – into equidimensional components.

Example 3 *We now consider the union of the space curve of example ?? and a surface $V(x^3 - y^2)$ which possesses a non-isolated singularity.*

```
> ring r=0,(x,y,z),dp;       // polynomial ring: char 0, 3 var.
> ideal I1=y2-xz,x2y-z2,x3-yz;  // the previously computed ideal
> ideal I2=x^3-y^2;          // the singular surface
> ideal I=intersect(I1,I2);  // the union of the two varieties
> I;
I[1]=x3y2-x4z-y4+xy2z
I[2]=x5y-x2y3-x3z2+y2z2
I[3]=x6-x3y2-x3yz+y3z
> LIB "primdec.lib";         // equidim. decomp. is in 'primdec.lib'
> list li=equidim(I);        // compute list of equidim. parts
> li;
li[1]:                       // ideal describing part of dim. 1
   _[1]=y2-xz
   _[2]=x2y-z2
   _[3]=x3-yz
li[2]:                       // ideal describing part of dim. 2
   _[1]=x3-y2
```

Using this equidimensional decomposition, we can then compute the singular locus of each of the equidimensional parts by the Jacobian criterion. The union

6

of these singular loci and of the intersection locus of the various parts now is precisely the singular locus of the whole variety.

Example 4 *(example* **??** *continued)*

```
> LIB "sing.lib";              // 'sing.lib' contains slocus
> ideal sL1=slocus(li[1]);     // singular locus of 1-dim. part
> ideal sL2=slocus(li[2]);     // singular locus of 2-dim. part
> sL2;
sL2[1]=x3-y2
sL2[2]=-2y
sL2[3]=3x2
> ideal inter12=li[1]+li[2];   // intersection of 1- and 2-dim. parts
> inter12;
inter12[1]=y2-xz
inter12[2]=x2y-z2
inter12[3]=x3-yz
inter12[4]=x3-y2
                               // union of contributions
                               // to sing. locus
> ideal sL=intersect(sL1,sL2,inter12);
> primdecGTZ(sL);
[1]:                           // first primary component:
   [1]:                        // singular locus of surface
      _[1]=y
      _[2]=x2
   [2]:
      _[1]=y
      _[2]=x
[2]:                           // second primary component:
   [1]:                        // singular locus of curve
      _[1]=z2                  // but also one of intersection
      _[2]=y2-yz               // points of the two parts
      _[3]=xyz
      _[4]=x2z
      _[5]=x3-yz
   [2]:
      _[1]=z
      _[2]=2y-z
      _[3]=x
[3]:                           // third primary component:
   [1]:                        // other intersection point
      _[1]=z-1                 // of the two parts
      _[2]=y-1
      _[3]=x-1
   [2]:
      _[1]=z-1
```

7

```
_[2]=y-1
_[3]=x-1
```

1.3 Finding the Correct Number of Components

As the following example shows, the result of the primary decomposition needs
to be interpreted with caution when counting the number of branches:

Example 5 *Consider the variety $V(\langle x^4 - yz^2, xy - z^3, y^2 - x^3z\rangle) \subset \mathbb{A}_\mathbb{C}$. Its
only singular point is the origin. The task is to compute the number of branches
of this space curve.*

```
> ring r=0,(x,y,z),dp;
> ideal I=x4-yz2,xy-z3,y2-x3z;        // the ideal of the curve
> primdecGTZ(I);                       // primary decomposition
[1]:
   [1]:
      _[1]=z8+yz6+y2z4+y3z2+y4
      _[2]=xz5+z6+yz4+y2z2+y3
      _[3]=-z3+xy
      _[4]=x2z2+xz3+xyz+yz2+y2
      _[5]=x3+x2z+xz2+xy+yz
   [2]:
      _[1]=z8+yz6+y2z4+y3z2+y4
      _[2]=xz5+z6+yz4+y2z2+y3
      _[3]=-z3+xy
      _[4]=x2z2+xz3+xyz+yz2+y2
      _[5]=x3+x2z+xz2+xy+yz
[2]:
   [1]:
      _[1]=-z2+y
      _[2]=x-z
   [2]:
      _[1]=-z2+y
      _[2]=x-z
```

*This result seems to imply that the number of branches would be 2. To
check the plausibility of this conclusion, we consider the Milnor number of the
singularity at the origin, which turns out to be 12. But by the formula $\mu =
2\delta - r + 1$, an even number of branches would imply an odd Milnor number.
Therefore, the conclusion that there are two branches is not plausible. The
reason for this is that we are calculating over the rationals, but thinking over the
complex numbers. In particular, the first primary component actually consists
of 4 components, as we see by considering the normalization.*[2]

---

[2]Recall that in our situation the normalization coincides with a parametrization of the
curve.

8

```
> LIB "normal.lib";              // normalization is in 'normal.lib'
> list li=normal(I);             // compute normalization
// 'normal' created a list of 1 ring(s).
// To see the rings, type (if the name of your list is nor):
     show( nor);
// To access the 1-st ring and map (similar for the others), type:
     def R = nor[1]; setring R;  norid; normap;
// R/norid is the 1-st ring of the normalization and
// normap the map from the original basering to R/norid
> size(li);                      // how many branches (over Q)
1
> def norring=li[1];             // consider branch more closely
> setring norring;
> basering;
//    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)^5-1
```

At first glance, it might seem strange that the ring describing the normalization has two variables, although it describes a 1-dimensional object. But looking at norid, we see that the second variable is only used to specify an appropriate field extension over the rationals such that all branches are separated. In particular, we see that we have 5 branches.

Another possibility to determine the correct number of branches, is the use of a projection to the plane and determine further data of the curve via Puiseux expansion (see ?? below).

2   Computing Invariants of Isolated Singularities

After discussing briefly some aspects of algorithmic calculations in local rings, this section contains a few examples of invariants which can be computed in practice. To each example, we also mention where further information on the algorithmic aspects can be found. This list of examples is by no means exhaustive, it is intended as a kind of appetizer for the audience to start discovering what is available as algorithmic tools for their field of research; for simplicity of the presentation, we only discuss examples whose implementation is also available in SINGULAR and not even half of the funtionality of SINGULAR in this area is mentioned.

9

2.1 Local and Global Considerations

Up to this point, we have only studied varieties, but not germs. As a consequence all computations have been performed in polynomial rings, not in power series rings.[3] Actually, a full implementation of power series rings on a computer is not feasable, but nevertheless many practical tasks can be tackled by using the localization of the respective polynomial ring at the origin instead[4].

To understand the basic idea behind the implementation of this type of localizations of polynomial rings, we first need to consider the problem of representation of polynomials on the computer. The need to represent polynomials on the computer in a unique way forces us to use a total ordering on the set of all monomials which has to be compatible with multiplication of monomials. If the monomial 1 is the smallest monomial, the monomial ordering is called global and the ring is a polynomial ring; if 1 is the largest monomial, the monomial ordering is called local and the ring is a localization of the polynomial ring at the origin. Orderings, in which some, but not all monomials are smaller than 1, are also possible and are usually referred to as mixed orderings. A detailed discussion of the influence of the choice of ordering on the ring is beyond the scope of this set of two talks and we refer the participants to a suitable textbook, e.g. [?].

Example 6 *In this example, we show some very simple calculations to illustrate the contrast between the local and global monomial orderings.*

The first of these small tasks is the calculation of a Gröbner basis resp. standard basis for the ideal of the variety consisting of the plane $V(z+1)$ and the two lines $V(x,y)$ and $V(x-1,y-1)$ in $\mathbb{A}^3_{\mathbb{C}}$:

```
> ring rg=0,(x,y,z),dp;          // polynomial ring in 3 var.: dp
> ring rl=0,(x,y,z),ds;          // localization at origin:    ds

> setring rg;                    // go back to polynomial ring
> ideal I1=z+1;                  // the plane V(z+1)
> ideal I2=x,y;                  // first line
> ideal I3=x-1,y-1;              // second line
> ideal Itemp=intersect(I1,I2);  // union of the first two
> ideal I=intersect(Itemp,I3);   // union of all three
> I;
I[1]=-xz+yz-x+y
I[2]=xyz+xy-yz-y

                                 // remark: I is radical by
                                 //         construction

> ideal J=groebner(I);           // compute Groebner basis
```

---

[3]The functionality of primary decomposition, radical and normalization is only available in polynomial ring, not in localizations thereof.

[4]For obvious reasons, the input and output still need to be specified in terms of polynomial data.

```
> J;
_[1]=xz-yz+x-y
_[2]=y2z+y2-yz-y

> setring rl;                         // now go to localization
> def I=imap(rg,I);                   // map ideal via identity map
> I;
I[1]=-x+y-xz+yz                       // observe the different way
I[2]=-y+xy-yz+xyz                     // of writing I[1]
> ideal J=groebner(I);                // compute standard basis
> J;
_[1]=x                                // we only see the components
_[2]=y                                // meeting the origin
```

Continuing with the same example, we now compute the dimesions and check whether the variety/germ is contained in the plane $V(x)$:

```
> setring rg;                         // back to polynomial ring
> dim(J);                             // 'dim' needs Groebner/standard basis
2                                     // dimension of the plane
> setring rl;                         // back to localization
> dim(J);                             // applying 'dim' at 0
1                                     // dimension of components meeting 0

> setring rg;
                                      // ideal membership test x in J ?
> reduce(x,J);                        // J needs to be Groebner/standard basis
x                                     // answer: no
>setring rl;
> reduce(x,J);                        // same question locally
0                                     // answer: yes
```

2.2   Dimension and Multiplicity

As we already used the dimension of a variety or a germ in the previous example, this seems to be a good moment to look at its calculation and at related data. The notion of dimension itself can be phrased in several ways (e.g. for a local ring $(R, \mathfrak{m})$: maximal length of chains of prime ideals, minimal number of generators of an m–primary ideal in a local ring $(R, \mathfrak{m})$,etc.), but most accessible to the use in practical calculations is the definition by means of the degree of the Hilbert-Samuel polynomial.

More precisely, it is possible to explicitly compute the Hilbert-Samuel polynomial of a given ideal with polynomial generators in the localization of a polynomial ring at the origin. The general idea of this calculation is to find a suitable system of generators (a standard basis of the ideal w.r.t. a local degree ordering), then pass to the ideal generated by the largest terms of the generators (the

so-called leading ideal) and compute the desired $\mathbb{C}$-vector space dimensions for this new (monomial) ideal in a combinatorial way. The degree and leading coefficient of this Hilbert-Samuel function yield the desired data. For an in depth discussion of this calculation and its theoretical background see e.g. chapter 5 of [?].

In SINGULAR, the dimension and multiplicity[5] are directly accessible as kernel commands `dim` and `mult`.

Example 7 *To illustrate the use of these commands, we now consider a space curve singularity at the origin consisting of an E_6 singularity in the x-y plane and the z-axis.*

```
> ring r=0,(x,y,z),ds;            // this is a local degree ordering
> ideal I=xz,yz,x3-y4;            // a space curve singularity
> I=groebner(I);                  // compute standard basis
> I;
I[1]=xz
I[2]=yz
I[3]=x3-y4
                                  // ideal generated by largest monomials of
> lead(I);                        // the generators of I
_[1]=xz
_[2]=yz
_[3]=x3
> dim(I);                         // compute dimension
1
> mult(I);                        // multiplicity
4
> dim(lead(I)),mult(lead(I));     // should give the same values
// ** _ is no standardbasis        // ** automatic warnings can be ignored
// ** _ is no standardbasis        // ** if ideal is monomial
1 4
```

2.3 Milnor and Tjurina Number

Another example, which illustrates the issue of local and global orderings nicely, is the calculation of the Milnor and Tjurina numbers of a given singularity. Recall that for an isolated hypersurface singularity defined by $f \in \mathbb{C}\{\underline{x}\}$, these invariants defined as the $\mathbb{C}$-vector space dimensions:

$$\mu = \dim_{\mathbb{C}} \left(\mathbb{C}\{\underline{x}\}/\langle \frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n} \rangle \right) \quad \tau = \dim_{\mathbb{C}} \left(\mathbb{C}\{\underline{x}\}/\langle f, \frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n} \rangle \right).$$

For more details on these invariants see any textbook on singularities, e.g. [?] in the curve case, [?] for hypersurfaces or [?] in the case of complete intersections.

---

[5]Calling the degree of the Hilbert-Samuel polynomial d, the multiplicity is $d!$ times leading coefficient of the Hilbert-Samuel polynomial.

Example 8 *In this example, we compute the Milnor and Tjurina number at the origin for the plane curve consisting of two cusps $V(x^2 - y^3)$ and $V(x^3 - y^2)$.*

```
> ring r=0,(x,y),ds;              // local ring in 2 variables
> ideal I=(x^2-y^3)*(x^3-y^2);    // the curve
> ideal Jac=jacob(I);             // jacobian ideal of I
> groebner(Jac);
_[1]=2x2y-5y4
_[2]=2xy2-5x4
_[3]=x5-y5
_[4]=y6
> vdim(groebner(Jac));            // the Milnor number
11
> vdim(groebner(Jac+I));          // the Tjurina number
10
```

Alternatively, we can also use the predefined commands in the library 'sing.lib':

```
> LIB "sing.lib";
> milnor(I);                      // the Milnor number
11
> tjurina(I);                     // the Tjurina number
10
```

But what would happen, if we specified a global ordering instead of the local ordering?

```
> ring r2=0,(x,y),dp;             // global ordering
> ideal I=(x^2-y^3)*(x^3-y^2);    // the curve
> ideal Jac=jacob(I);             // jacobian ideal of I
> groebner(Jac);
_[1]=3x2y3-5x4+2xy2
_[2]=3x3y2-5y4+2x2y
_[3]=x5-y5
_[4]=9y7-19x4y+10xy3
> vdim(groebner(Jac));
21
> vdim(groebner(Jac+I));
15
```

The numbers, which we computed here, are precisely the sums over the Milnor resp. Tjurina numbers of all singularities of the affine curve. Therefore we expect to find further critical points outside the origin whose multiplicities add up to 10 resp. further singular points whose Tjurina numbers add up to 5. To check this, we determine the singular locus, move to each of the other singular points and compute Milnor and Tjurina numbers there. Subsequently, we also study the critical locus, which, of course, contains the singular locus.

13

```
> LIB "primdec.lib";
> minAssGTZ(slocus(I));              // components of singular locus
[1]:                                 // the origin -- we knew that one
   _[1]=y
   _[2]=x
[2]:                                 // the point (1,1)
   _[1]=y-1
   _[2]=x-1
[3]:                                 // a set of 4 points
   _[1]=y4+y3+y2+y+1                  // <--- keep this in mind (*)
   _[2]=y3+y2+x+y+1

> setring r;                         // go back to local ring
> map m1=r2,x+1,y+1;                  // translation of (1,1) to origin
> def I2=m1(I);                      // move our curve, to study at (1,1)
> I2;
I2[1]=6x2-13xy+6y2+9x3-11x2y-11xy2+9y3+5x4-3x3y-10x2y2-3xy3+5y4
     +x5-3x3y2-3x2y3+y5-x3y3
> milnor(I2);                        // Milnor number
1
> tjurina(I2);                       // Tjurina number
1

                                     // extend basefield to look
                                     // at the 4 points
> ring r2a=(0,a),(x,y),ds;           // adjoining parameter a
> minpoly=a4+a3+a2+a+1;              // minimal polynomial, see (*)

> map m2=r2,x-a3-a2-a-1,y+a;         // go to one of the points
> def I3=m2(I);                      // map the curve
> milnor(I2);                        // Milnor number
1
> tjurina(I2);                       // Tjurina number
1

> setring r2;                        // go back to r2
> minAssGTZ(jacob(I));               // decompose set if critical points
[1]:                                 // origin -- already considered
   _[1]=y
   _[2]=x
[2]:                                 // (1,1) -- already considered
   _[1]=y-1
   _[2]=x-1
[3]:                                 // 4 points -- already considered
   _[1]=y4+y3+y2+y+1
   _[2]=y3+y2+x+y+1
```

14

```
[4]:                              // 4 critical points
   _[1]=81y4+54y3+36y2+24y+16
   _[2]=27y3+18y2+12x+12y+8
[5]:                              // 1 critical point
   _[1]=3y-2
   _[2]=3x-2
```

Actually, it would not have been necessary to move to each of the points and check the Milnor and Tjurina numbers explicitly, because we only had a difference of 10 for the Milnor and of 5 for the Tjurina number and this equals the number of additional points in the critical resp. singular locus.

The Milnor and Tjurina numbers for isolated complete intersection singularities are available by the same command – in the case of the Milnor number by use of the Lê-Greuel formula. The Tjurina number for Cohen-Macaulay codimension 2 singularities, which are not ICIS, is provided in the library 'spcurve.lib'; in the general case it can be obtained via the command T^1, see below.

2.4 Puiseux Expansion

To determine further invariants of the plane curve in the previous example, we are now going to use Puiseux expansion. More precisely, we are going to use Hamburger-Noether expansion, an analogue to Puiseux expansion which works in arbitrary characteristic. More details on Hamburger-Noether expansion can be found in [?].

Example 9 *Continuing where we stopped in our calculations in the previous example, we now apply Hamburger-Noether expansion and extract information about the given plane curve from it. Recall that this curve consisted of two branches $V(x^2 - y^3)$ and $V(y^2 - x^3)$.*

```
> LIB "hnoether.lib";           // load Hamburger-Noether library
> poly f=I[1];                  // hnexpansion needs argument of type poly
> hnexpansion(f);               // call Hamburger-Noether expansion
[1]:                            // result lives in a new ring
   //    characteristic : 0
   //    number of vars : 2
   //         block   1 : ordering ls
   //                   : names    x y
   //         block   2 : ordering C
> def S=_[1];                   // give that ring the name S
> setring S;                    // and change to it
> hne;                          // result can be found in hne
[1]:                            // technical data, not really readable
   [1]:
      _[1,1]=0
      _[1,2]=x
      _[1,3]=0
```

15

```
                    _[2,1]=0
                    _[2,2]=1
                    _[2,3]=x
          [2]:
              1,2
          [3]:
              0
          [4]:
              0
   [2]:
      [1]:
              _[1,1]=0
              _[1,2]=x
              _[1,3]=0
              _[2,1]=0
              _[2,2]=1
              _[2,3]=x
          [2]:
              1,2
          [3]:
              1
          [4]:
              0
> displayHNE(hne);                     // the better way to look at it ;-)
// Hamburger-Noether development of branch nr.1:
HNE[1]=-y+z(0)*z(1)
HNE[2]=-x+z(1)^2


// Hamburger-Noether development of branch nr.2:
HNE[1]=-x+z(0)*z(1)
HNE[2]=-y+z(1)^2
                                       // Caution!
                                       // numbering of branches may
                                       // change when calling hnexpansion
                                       // a second time on the same input


> displayInvariants(hne);              // the invariants computed from it
 --- invariants of branch number 1 : ---
 characteristic exponents  : 2,3
 generators of semigroup   : 2,3
 Puiseux pairs             : (3,2)
 degree of the conductor   : 2
 delta invariant           : 1
 sequence of multiplicities: 2,1,1

 --- invariants of branch number 2 : ---
```

16

```
characteristic exponents  : 2,3
generators of semigroup   : 2,3
Puiseux pairs             : (3,2)
degree of the conductor   : 2
delta invariant           : 1
sequence of multiplicities: 2,1,1


-------------- contact numbers : --------------

branch |   2
-------+-----
   1 |   1


-------------- intersection multiplicities : --------------

branch |   2
-------+-----
   1 |   4


-------------- delta invariant of the curve :   6
```

2.5 Classification of Hypersurface Singularities

Sometimes, we want to check whether a given singularity is in Arnold's list of
hypersurface singularities [?]. This test is implemented in SINGULAR as well:

Example 10 *Still continuing with the singularity which we have been considering in the previous examples, we now use the Arnold-classifier to determine
its type:*

```
> LIB "classify.lib";                  // classifier library
> setring r;                           // need to be in correct ring
> def f=I[1];                          // input needs to be of type 'poly'
> quickclass(f);                       // first guess via invariants
Singularity R-equivalent to :  Z[k,12k+6r-1]=Z[1,11] Y[k,r,s]=Y[1,1,1]
Hilbert-Code of Jf^2
We have  2 cases to test
null form
[1]:
    Z[k,12k+6r-1]=Z[1,11] Y[k,r,s]=Y[1,1,1]
[2]:
   2
                                       // classification following
> classify(f);                         // Arnold's algorithm
About the singularity :
        Milnor number(f)   = 11
```

17

```
            Corank(f)           = 2
            Determinacy        <= 8
Guessing type via Milnorcode:    Z[k,12k+6r-1]=Z[1,11] Y[k,r,s]=Y[1,1,1]

Computing normal form ...
   Arnold step number 16
The singularity
   -x2y2+x5+y5-x3y3
is R-equivalent to Y[1,p,q] = T[2,4+p,4+q].
   Milnor number = 11
   modality      = 1
```

2.6 Monodromy and Spectral Numbers

Another set of invariants which has been made accessible to practical computations in recent years are the monodromy and spectral numbers.[6]

Example 11 *Let us consider the example of the isolated hypersurface singularity defined by the polynomial $f = x^5 + y^5 + x^2y^2$. We first want to compute a matrix M such that $e^{-2\pi i M}$ is the monodromy matrix of the given f:*

```
> LIB "gmssing.lib";              // monodromy, spectrum etc.
> ring r=0,(x,y),ds;              // as usual first the 'ring' definition
> poly f=x5+y5+x2y2;              // then the polynomial
> monodromy(f);                   // compute data of the monodromy:
[1]:                              // eigenvalues of M
   _[1]=1/2
   _[2]=7/10
   _[3]=9/10
   _[4]=1
   _[5]=11/10
   _[6]=13/10
[2]:                              // sizes of blocks
   2,1,1,1,1,1
[3]:
   1,2,2,1,2,2                    // multiplicities
```

---

[6]See Ebeling's and Steenbrink's series of talks at this School for definitions and properties of the monodromy and spectrum.

18

Therefore, the Jordan normal form of M has the following structure:

$$\begin{pmatrix} \frac{1}{2} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{7}{10} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{7}{10} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{9}{10} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{9}{10} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{11}{10} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{11}{10} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{13}{10} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{13}{10} \end{pmatrix}.$$

The same library also provides support for calculation of the spectral numbers of f using standard basis methods for the microlocal structure of the Brieskorn lattice. For details on this algorithmic approach and on more sophisticated data which can also be acquired along these lines see [?].

```
> spectrum(f);                        // compute the spectrum
[1]:                                  // spectral numbers
   _[1]=-1/2
   _[2]=-3/10
   _[3]=-1/10
   _[4]=0
   _[5]=1/10
   _[6]=3/10
   _[7]=1/2
[2]:                                  // their multiplicities
   1,2,2,1,2,2,1
> spprint(_);                         // pretty printing of previous output
(-1/2,1),(-3/10,2),(-1/10,2),(0,1),(1/10,2),(3/10,2),(1/2,1)
```

3 Deformations of Singularities

After using computational methods for studying the singular locus of a variety and for determining invariants of isolated singularities, we now turn our interest to families of singularities. More precisely, we first consider how to compute the T^1 and T^2 of an isolated singularity, construct versal families and then proceed to a more detailed study of certain special families of singularities.[7]

3.1 T^1 and T^2

In the case of a isolated complete intersection singularity $(X,0)$ given by $I \subset \mathbb{C}\{x_1,\ldots,x_n\} = \mathcal{O}_n$, it is a well known fact that there are no obstructions to

[7]Definitions and properties of T^1, T^2 and versal deformations can e.g. be found in the textbook [?].

lifting first order deformations. Hence we only need to compute T^1.

By definition, the T^1 is computed, in general, by determining the normal module $N_{X,0} = Hom_{\mathcal{O}_n}(I, \mathcal{O}_{X,0})$ and then passing to the cokernel of the map from the free module of $\mathcal{O}_n$-derivations to $N_{X,0}$ defined by

$$
\begin{aligned}
\theta &\longrightarrow N_{X,0} \\
\vartheta &\longmapsto (f \mapsto \vartheta(f))
\end{aligned}
$$

This general approach to comuting $T^1_{X,0}$ can get rather time consuming. For complete intersections, however, we know the normal module and the cokernel of the map explicitly[8]:

Example 12 *As a complete intersection example. let us consider the isolated space curve singularity defined by the ideal $I = \langle x^2 + y^2 + z^3, yz \rangle$.*

```
> ring r=0,(x,y,z),ds;
> ideal I=x2+y2+z3,yz;             // the singularity
> def N=I*freemodule(2);           // presentation of normal module
> def T=jacob(I)+N;                // presentation of T^1
> vdim(std(T));                    // the Tjurina number
6
> kbase(std(T));                   // base of 1st order miniversal deform.
_[1]=z2*gen(1)
_[2]=z*gen(1)
_[3]=gen(1)
_[4]=z*gen(2)
_[5]=x*gen(2)
_[6]=gen(2)
```

This is, of course also available as a SINGULAR *command:*

```
> LIB "sing.lib";                  // command is in 'sing.lib'
> Tjurina(I);                      // compute T1, ICIS case
// Tjurina number = 6
_[1]=x*gen(1)
_[2]=y*gen(2)+3z2*gen(1)
_[3]=2y*gen(1)+z*gen(2)
_[4]=x2*gen(2)+y2*gen(2)+z3*gen(2)
_[5]=xz*gen(2)
_[6]=z2*gen(2)
_[7]=z3*gen(1)
> kbase(std(_));                   // base of 1st order miniveral deform.
_[1]=z2*gen(1)
```

---

[8]In the case of Cohen-Macaulay codimension 2 singularities, there is a direct method for computing these data, too. Whenever there is such a direct approach, it tends to be much more efficient than the general one and hence should be preferred.

20

```
_[2]=z*gen(1)
_[3]=gen(1)
_[4]=z*gen(2)
_[5]=x*gen(2)
_[6]=gen(2)
```

In the general case, we do not only need the T^1, whose calculation was already explained above, but also the obstructions to lifting first order deformations to deformations: Given an isolated singularity $(X, 0)$ defined by the ideal $I = \langle f_1, \ldots, f_k \rangle \subset \mathcal{O}_n$, we denote the module of syzygies of I by $\mathcal{R}$ and the submodule of Koszul relations by $\mathcal{R}_0$. Then the T^2 can be computed as

$$T^2 := Hom_{\mathcal{O}}(\mathcal{R}/\mathcal{R}_0, \mathcal{O}_{X,0})/Hom_{\mathcal{O}}(\mathcal{O}^k, \mathcal{O}_{X,0}).$$

Example 13 *As an example in the general case, let us consider the isolated singularity defined by the 2-minors of the matrix*

$$\begin{pmatrix} x & y & z & u \\ y & z & u & v \end{pmatrix}$$

and compute its T^1 and T^2 using the appropriate built-in commands of SINGU-LAR.

```
> LIB "sing.lib";              // T1 and T2 are in 'deform.lib'
> ring r1=0,(x,y,z,u,v),ds;    // local ring in 5 variables
> matrix M[2][4] = x,y,z,u,y,z,u,v; // the matrix, see above
> ideal I=minor(M,2);          // the ideal
> I;
I[1]=-u2+zv
I[2]=-zu+yv
I[3]=-yu+xv
I[4]=z2-yu
I[5]=yz-xu
I[6]=-y2+xz
> T_12(I);                     // compute T1 and T2
// dim T_1 = 4
// dim T_2 = 3
[1]:                           // standard basis of T^1
   _[1]=gen(8)+2*gen(4)
   _[2]=gen(7)
   _[3]=gen(6)+gen(2)
   _[4]=gen(5)+gen(1)
   _[5]=gen(3)
   _[6]=x*gen(9)
   _[7]=2x*gen(4)+z*gen(2)
   _[8]=x*gen(2)
   _[9]=x*gen(1)+y*gen(2)
```

21

```
  _[10]=y*gen(9)+z*gen(2)
  _[11]=2y*gen(4)-z*gen(1)+u*gen(2)
  _[12]=y*gen(2)
  _[13]=y*gen(1)+z*gen(2)
  _[14]=z*gen(9)+u*gen(2)
  _[15]=2z*gen(4)-u*gen(9)-u*gen(1)
  _[16]=z*gen(2)
  _[17]=3z*gen(1)-u*gen(2)
  _[18]=u*gen(9)+3u*gen(1)
  _[19]=2u*gen(4)-v*gen(9)-v*gen(1)
  _[20]=u*gen(2)
  _[21]=2u*gen(1)-v*gen(2)
  _[22]=v*gen(9)+v*gen(1)
  _[23]=v*gen(4)
  _[24]=v*gen(2)
  _[25]=v*gen(1)
[2]:                                  // standard basis of T2
  _[1]=gen(9)
  _[2]=gen(7)+gen(5)
  _[3]=gen(6)
  _[4]=gen(3)
  _[5]=gen(2)
  _[6]=gen(1)
  _[7]=x*gen(8)
  _[8]=x*gen(5)
  _[9]=x*gen(4)
  _[10]=y*gen(8)-z*gen(5)-u*gen(4)
  _[11]=y*gen(5)+z*gen(4)
  _[12]=y*gen(4)
  _[13]=z*gen(8)
  _[14]=z*gen(5)+u*gen(4)
  _[15]=z*gen(4)
  _[16]=u*gen(8)
  _[17]=u*gen(5)+v*gen(4)
  _[18]=u*gen(4)
  _[19]=v*gen(8)
  _[20]=v*gen(5)
  _[21]=v*gen(4)
```

Being able to compute T^1 and T^2 explicitly, it is a natural question to ask whether we can also determine versal deformations up to a given degree in practice. The answer is affirmative and the corresponding algorithm is implemented in the library deform.lib.

```
> LIB "deform.lib";
> list L=versal(I,5);                // compute versal deformation up to degree 5
```

22

```
> L;                            // result is list of rings
// ready: T_1 and T_2
// start computation in degree 2.


// 'versal' returned a list, say L, of four rings. In L[1] are stored:
//    as matrix Fs: Equations of total space of the miniversal deformation,
//    as matrix Js: Equations of miniversal base space,
//    as matrix Rs: syzygies of Fs mod Js.
// To access these data, type
     def Px=L[1]; setring Px; print(Fs); print(Js); print(Rs);

[1]:
   //    characteristic : 0
//   number of vars : 9
//        block   1 : ordering ds
//                  : names    A B C D
//        block   2 : ordering ds
//                  : names    x y z u v
//        block   3 : ordering C
[2]:
   //    characteristic : 0
//   number of vars : 9
//        block   1 : ordering ds
//                  : names    A B C D
//        block   2 : ordering ds
//                  : names    x y z u v
//        block   3 : ordering C
// quotient ring from ideal ...
[3]:
   //    characteristic : 0
//   number of vars : 4
//        block   1 : ordering ds
//                  : names    A B C D
//        block   2 : ordering C
[4]:
   //    characteristic : 0
//   number of vars : 9
//        block   1 : ordering ds
//                  : names    A B C D
//        block   2 : ordering ds
//                  : names    x y z u v
//        block   3 : ordering C
// quotient ring from ideal ...
> def R1=L[1];
> setring R1;                   // go to first of the returned rings
```

23

```
> Js;                                   // equations of miniversal base space
Js[1,1]=BD
Js[1,2]=-AD+D2
Js[1,3]=-CD
> Fs;                                    // equations of miniveral total space
Fs[1,1]=-u2+zv+Bu+Dv
Fs[1,2]=-zu+yv-Au+Du
Fs[1,3]=-yu+xv+Cu+Dz
Fs[1,4]=z2-yu+Az+By
Fs[1,5]=yz-xu+Bx-Cz
Fs[1,6]=-y2+xz+Ax+Cy
```

3.2   Studying Families of Singularities

Having constructed versal families in the previous example, we now proceed
to study the question of stratifying the base space of a certain classes of fami-
lies of singularities with respect to the Tjurina number. This question can be
dealt with algorithmically for versal families of simple hypersurface and Cohen-
Macaulay codimension 2 singularities and for families of semiquasihomogeneous
singularities (again hypersurfaces or CM codimension 2) with fixed initial part.
In the first case, it can be used as one ingredient when determining an ad-
jacency to another singularity explicitly; in the second case, it is one step in
the construction of moduli spaces for semiquasihomogeneous singularities with
fixed initial part (for more details on this topic see e.g. [?]). We only consider
the first situation here, as the latter one involves the use of a rather technical
modification of the standard basis algorithm.

Example 14 *To keep the calculations as simple as possible, we only consider
a very small example, an A_3–singularity. We first compute a versal family by
means of calculation of a vector space basis for the T^1 (Tjurina algebra) and
the relative T^1 of this family:*

```
> ring r=0,(x,y),ds;
> poly f=x^4+y^2;                        // the singularity
> ideal kb=kbase(Tjurina(f));            // the vector space basis for T1
> kb;
kb[1]=x2
kb[2]=x
kb[3]=1
> ring rt=0,(a,b,c,x,y),ds;              // suitable ring for total space
> poly F=x^4+y^2+a+b*x+c*x^2;            // versal family
> ideal jF=diff(F,x),diff(F,y),F;        // presentation of relative T1
                                         // but as module over ring rt,
                                         // we need it as C[a,b,c]-module

> jF;
jF[1]=b+2cx+4x3
```

24

```
jF[2]=2y
jF[3]=a+bx+y2+cx2+x4
```

We know that the jF is a finitely presentable K[a,b,c] module. As f is a hy-persurface singularity, we can determine the corresponding presentation matrix by looking at the Euler relation and suitable products of it with monomials in x and y. (In this example only the products with x and x^2 are relevant.)

```
                                        // suitable ring for finding Euler rel.
> ring rg=0,(x,y,a,b,c),(dp(2),dp); // (Q[a,b,c])[x,y]

> def jF=imap(rt,jF);                   // fetch jF from rt
> jF=mstd(jF)[2];                       // find minial system of generators
                                        // for jF
> jF;                                   // look at jF
jF[1]=y
jF[2]=4x3+2xc+b
jF[3]=2x2c+3xb+4a                       // <-- Euler relation
> matrix Tmat[3][3];
> def tempmat=coef(jF[3],xy);           // give temporary name, because lists
> Tmat[1,1..3]=tempmat[2,1..3];         // can only be formed from named objects
> tempmat=coef(reduce(jF[3]*x,jF[2]),xy);
> Tmat[2,1..3]=tempmat[2,1..3];
> tempmat=coef(reduce(jF[3]*x^2,jF[2]),xy);
> Tmat[3,1..3]=tempmat[2,1..3];
> print(Tmat);                          // presentation matrix of T1
                                        // as Q[a,b,c]-module
2c,     3b,     4a,
3b,     -c2+4a,-1/2bc,
-c2+4a,-2bc,   -3/4b2
```

The strata of constant Tjurina number can now be obtained by means of the flattening stratification of the relative T^1. This implies that we need to determine the Fitting ideals of the module, that is we need to determine the minors of size one, two and three:

```
> ideal min1=mstd(minor(Tmat,1))[2]; // minimal set of generators for 1-minors
> min1;
min1[1]=c
min1[2]=b
min1[3]=a
> ideal min2=mstd(minor(Tmat,2))[2]; // dito for 2-minors
> min2;
min2[1]=2c3+9b2-8ac
min2[2]=bc2+12ab
min2[3]=3b2c-8ac2+32a2
> ideal min3=mstd(minor(Tmat,3))[2]; // and for 3-minors
```

25

```
> min3;
min3[1]=4b2c3-16ac4+27b4-144ab2c+128a2c2-256a3
```

From this computation we can see that the maximal value of the Tjurina number is attained exactly for the fiber over the point $V(a, b, c)$ of the base. For fibres over points outside of $V(4b^2c^3 - 16ac^4 + 27b^4 - 144ab^2c + 128a^2c^2 - 256a^3)$, which is the swallowtail singularitiy, on the other hand there are no singularities. The Tjurina number is 2 for points in $V(\mathtt{min2}) \setminus V(a, b, c)$. It is 1 for points in $V(\mathtt{min3}) \setminus V(\mathtt{min2})$, i.e. points on the swallowtail which do not lie on the curve $V(\mathtt{min2})$.
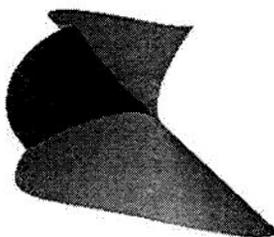
Figure 1: The swallowtail singularity $V(\mathtt{min3})$.
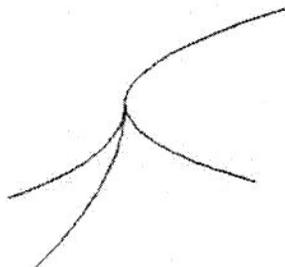
Figure 2: The singular locus of the swallowtail singularity $V(\mathtt{min2})$

26

4 Varieties with Singularities

In this last section, we consider two areas of more complex applications of computational methods in singularity theory: the task of finding hypersurfaces with prescribed singularities and the task of resolution of singularities. In the first case, the goal is more of theoretical nature and explicit calculations are basically used to check whether certain conditions are satisfied or for finding good examples which show certain properties. In the second case, the set-up is very different: The task itself is computational, but it consists of many different computational aspects each of which needs to be treated carefully in order to obtain an efficient implementation.

4.1 Hypersurfaces with Prescribed Singularities

Here, we briefly sketch two applications of computer algebra tools in this area: first we treat the question of finding an upper bound for how many singularities of a given type can fit on a hypersurface of a given degree, then we outline how computational tools aided in the search for examples of surfaces of fixed degree with a high number of double points.

Example 15 *The question, which we are treating in this example, is the following: What is the maximal number of singularities of type $T_{3,3,3}$ that can occur on a surface of degree 7 in $\mathbb{P}^3$?*
Let us fist recall that the singularities of type $T_{3,3,3}$ form a μ-constant 1-parameter family given by equations of the kind

$$x^3 + y^3 + z^3 + t \cdot xyz = 0, \quad \text{where} \quad t^3 \neq -27.$$

To obtain the desired bound, we now use the semicontinuity property of the spectrum. More precisely, the number of spectral numbers of the singularities of a deformation of a given hypersurface in an intervall $(a, a+1]$ cannot exceed the number of spectral numbers of the original singularity in this intervall; for semiquasihomogeneous singularities the same statement also holds for the intervalls $(a, a+1)$.

```
> LIB "gmssing.lib";              // here are the spectrum related commands
> ring R=0,(x,y,z),ds;            // local ring in 3 variables
> poly f=x^3+y^3+z^3;             // a singularity of type T_3,3,3
> list s1=spectrum(f);            // compute its spectrum
> s1;
[1]:                             // spectral numbers
   _[1]=0
   _[2]=1/3
   _[3]=2/3
   _[4]=1
[2]:                             // multiplicities
   1,3,3,1
```

27

```
> poly g = x^7+y^7+z^7;           // any surface of degree 7 is a
                                   // deformation of this surface
> list s2 = spectrum(g);          // compute its spectrum, takes longer
> s2;
[1]:                              // spectral numbers
   _[1]=-4/7
   _[2]=-3/7
   _[3]=-2/7
   _[4]=-1/7
   _[5]=0
   _[6]=1/7
   _[7]=2/7
   _[8]=3/7
   _[9]=4/7
   _[10]=5/7
   _[11]=6/7
   _[12]=1
   _[13]=8/7
   _[14]=9/7
   _[15]=10/7
   _[16]=11/7
[2]:                              // multiplicities
   1,3,6,10,15,21,25,27,27,25,21,15,10,6,3,1
> spsemicont(s2,list(s1));        // checking semicontinuity condition
[1]:
   18
> spsemicont(s2,list(s1),1);      // checking sqh.semicont. condition
[1]:
   17
```

Thus a septic in $\mathbb{P}^7$ can at most contain 17 singularities of type $T_{3,3,3}$.

On the other hand, computer algebra methods have recently been sucessfully used by O. Labs and D. van Straten to construct a septic with 99 nodes [?].[9] The basic idea behind the approach of Labs and van Straten is the following: They start with a 7-parameter family of septics and develop conditions to easily determine the number of nodes on a given septic from a 5-parameter subfamily of this family. Then they pass to small prime fields ($11 \le p \le 53$) and explicitly check the actual number of nodes on the septic for all possible parameter combinations to obtain those which provide exactly 99 nodes. Further geometric considerations in characteristic zero lead to a condition for the parameters which can be described as the zero locus of a single univariate polynomial of

---

[9]Up to degree 6 the maximal number of nodes on a surface is known, that is there are known examples possessing exactly the number of of nodes specified by an upper bound. In degree 7, however, Varchenko's spectrum bound and Giventhal's bound both lead to an upper bound of 104 for the number of nodes on a septic, the septic with the highest number of nodes that had been known prior to the example of Labs and van Straten had 93 nodes.

degree 150, which is, of course, still too large to be of any direct use. Therefore they factorize the polynomial and plug into each of the factors the solutions which were previously obtained over the small prime fields. This leads to only one factor of degree 3 whose vanishing locus contains one real solution; it can then be checked by explicit calculation that the surface corresponding to this parameter has precisely 99 nodes and no other singularities.
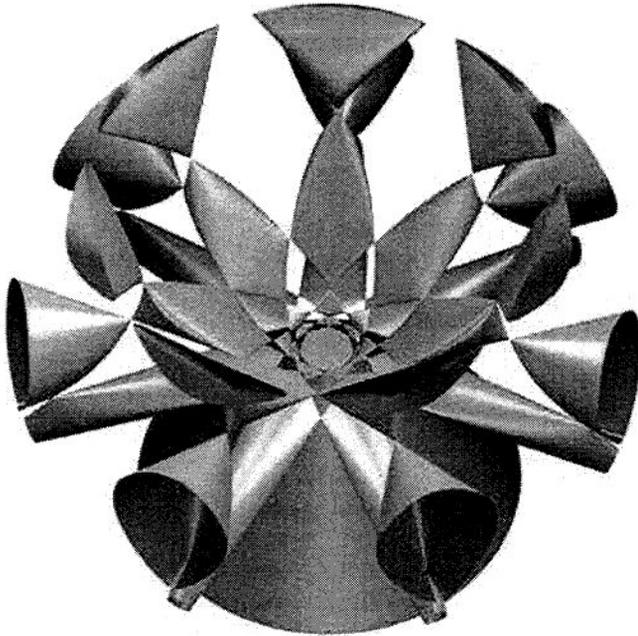

Figure 3: The septic surface with 99 real nodes found by O. Labs and D. van Straten.

4.2 Resolution of Singularities

The last aspect of computational methods for studying singularities is the task of resolution of singularities. As the series of talks of H. Hauser at this school is devoted to the theoretical background of this topic, we only consider the practical side of it.

As the process of resolution of singularities consists of a sequence of blow-ups at suitable centers, the computational goal can be decomposed into two separate tasks: the computation of a blow-up at a given center and the search for suitable centers. This latter task itself is the key point of the algorithm and a detailed explanation of its construction can be found in H. Hauser's notes; from the computational point of view the main difficulties of it are the descent in dimension and the computation of the operator Δ which is used to determine

the locus of maximal order.

One issue, which needs to be discussed before focusing on these three tasks, is the representation of the data on the computer. Because we are not just dealing with affine varieties, it is necessary to cover the objects, which are considered, by charts. In particular, each blow-up introduces a new covering by charts in the usual way. Hence the whole resolution process leads to a tree of charts which introduces a new difficulty for working with the result in explicit examples: the identification of points which appear in several charts.

Based on the implementation of the resolution process, it is then possible to determine resolution related invariants explicitly. As an example, we discuss how to determine the intersection matrix of the exceptional divisors in a (non-embedded) resolution of a surface. For a detailed discussion of practical aspects of other applications see [?].

4.2.1 Blowing Up

The blow-up can be implemented as a preimage computation. More precisely, a blow-up $\pi : W_1 \longrightarrow W$ at a smooth center C can be computed as follows: the ideal $I(W_1)$ is determined as the preimage of $\mathcal{I}(W)$ under the map

$$
\begin{array}{rcl}
\phi : \mathbb{C}[x_1, \ldots, x_n, y_0, \ldots, y_{s-1}] & \longrightarrow & \mathbb{C}[x_1, \ldots, x_n, t] \\
x_i & \longmapsto & x_i \\
y_i & \longmapsto & t \cdot g_i
\end{array}
$$

where the variables $x_1, \ldots, x_n$ are the original ones, $y_0, \ldots, y_{s-1}$ new variables and $g_0, \ldots, g_{s-1} \in \mathbb{C}[x_1, \ldots, x_n]$ a set of generators of the ideal of C. The blown-up space W_1 is covered by the affine charts $D(y_i)$. Total transforms of subvarieties of W can then be computed in the usual way and passing to weak respectively strict transforms can be implemented as iterated ideal quotients $((J : I(H)) \cdots : I(H))$ where J denotes the ideal of the total transform of the subvariety and $I(H)$ the ideal of the new exceptional divisor.

Obviously, the difficulty of the computation of the preimage depends very much on the generators g_i of the center and on the total number of variables involved. In particular, successive blowing-ups in smooth irreducible centers turn out to be by far less expensive than blowing-up at several smooth (disjoint) irreducible centers simultaneously. Therefore we always apply primary decomposition of the center and blow up at each of the components separately. Clearly, this is possible because a blow-up is an isomorphism outside of the center. The draw-back of this improvement is the fact that more charts are produced and hence more duplicate calculations can occur in future steps of the resolution process; but this tradeoff still pays off in a very large number of practical applications.

Another enhancement to the resolution process follows from the fact that not all s charts arising from a single blow-up contain new information. It may very well happen that in one or more charts we do not see any new information that is not already provided by the other charts. In this case, such charts may

be dropped. We have been very careful not to state what the information is, because that can depend very much on the data that is to be computed from the resolution: If the goal is, e.g., to compute the intersection matrix of the exceptional divisors of a resolved surface, the relevant information, which needs to be kept, just consists of the points of the transform of the surface. If on the other hand, the goal is the computation of a ζ-function, no charts may be dropped.

4.2.2 Computing the Order

Let us first recall the definition of the order of an ideal and of Δ:
Let W be a non-singular algebraic variety, J a sheaf of ideals in $\mathcal{O}_W$ and $w \in W$ a closed point. Then the order at w with respect to J is defined as

$$v_J(w) = \sup\{m \mid J_w \subseteq \mathfrak{m}_{W,w}^m\}.$$

We define $\Delta(J) \subset \mathcal{O}_W$ as the sheaf of ideals which is locally generated by

$$\{g_i | 1 \le i \le s\} \cup \{\frac{\partial g_i}{\partial x_j} | 1 \le i \le s, 1 \le j \le d\},$$

where $x_1, \ldots, x_d$ is a regular system of parameters for $\mathcal{O}_{W,w}$ and $g_1, \ldots, g_s$ are a set of generators for J_w. $\Delta^i(J)$ is then inductively defined as $\Delta(\Delta^{i-1}(J))$. Recall further that the locus of order at least b of J coincides with $V(\Delta^{b-1}(J))$.

The definitions of the order and of $\Delta(J)$ heavily rely on using generators for the ideal $J \subset \mathcal{O}_{W,w}$ and a regular system of parameters for $\mathcal{O}_{W,w}$ at the given closed point $w \in W$. Theoretically this is fine, but in practice it is, of course, not feasible to compute at each point of W. Here, the use of a set of generators of $J\mathcal{O}_{W,w}$ does not cause any problems, since we are working on affine charts U_i and on each chart we are specifying J by a set of generators anyway. The difficulties arise from the need for a global system of local regular parameters for $\mathcal{O}_{W,w}$ on U_i, which, in general, does not exist. Instead it is necessary to pass to a suitable open covering $\{U_{ij}\}$ of U_i such that for each U_{ij} we can find a global system giving rise to a regular system of parameters at each point of U_{ij}. This, in turn, increases the number of charts which we can avoid by recombining the results on the U_{ij} to one on U_i. More precisely, $\Delta(J)$ is determined by the following algorithm:

Algorithm Delta

Input $(g_1, \ldots, g_r)$ generating $\mathcal{I}_W \subset \mathbb{C}[x_1, \ldots, x_n] = \mathcal{O}_{U_i}$
$(f_1, \ldots, f_s)$ generating $\mathcal{I}_X \subset \mathbb{C}[x_1, \ldots, x_n]$
such that $V(\mathcal{I}_W)$ is equidimensional and regular and $\mathcal{I}_W \subset \mathcal{I}_X$.

Output $\Delta(\mathcal{I}_X) \subset \mathbb{C}[x_1, \ldots, x_n] = \mathcal{O}_{U_i}$

1. if $\mathcal{I}_W = (0)$
   then return$((f_1, \ldots, f_s, \frac{\partial f_1}{\partial x_1}, \ldots, \frac{\partial f_s}{\partial x_k}))$

31

2. Initialization
   $C = \{f_1, \dots, f_s\}$
   $D = (1)$
   $L1 = \{\; n - \dim(W)$ square submatrices of the Jacobian matrix
         of $\mathcal{I}_W$ whose determinant is non-zero$\}$ [10]

3. **while** $(L1!=\emptyset)$

   - choose $M \in L1$
     $L1 = L1 \setminus \{M\}$
   - $q = \det(M)$
   - determine an $n - \dim(W)$ square matrix A such that
     $A \cdot M = q \cdot E_{n-\dim(W)}$ [11]
   - determine components of $\Delta(J)$ not lying inside $V(q)$:

   $$C_M = C \cup \left\{ q \cdot \frac{\partial f_i}{\partial x_j} - \sum_{\substack{k \text{ row of } M \\ l \text{ column of } M}} \frac{\partial g_l}{\partial x_j} A_{lk} \frac{\partial f_i}{\partial x_k} \;\middle|\; \begin{array}{l} 1 \leq i \leq s, \\ j \text{ not row of } M \end{array} \right\}$$

   $C_M = sat(C_M, q)$

   - Add these components to the previously found ones:
     $D = D \cap C_M$

4. return(D)

The basic idea behind this algorithm is that W is regular and hence at each point there is at least one $(n - \dim(W))$-minor of the Jacobian matrix of $\mathcal{I}_W$ which does not vanish. So $\Delta(J)$ is computed separately on each complement of a minor of the Jacobian, then we pass to all of U_i again by saturation and combine the results of the computations on the complements.

From the practical point of view the above algorithm still needs to be improved to avoid redundant calculations. In particular, one should first check whether there is a minor of the appropriate size which is itself an element of $\mathbb{C}$. In this case, the complement of the minor is the whole open set U_i and the other minors do not give any new contributions.

4.2.3 Descent in Dimension

For the descent in dimension, that is the computation of the Coeff-ideal, the crucial point is the choice of the smooth hypersurface Z which is subject to two normal crossing conditions regarding the exceptional divisors. As soon as such a hypersurface is found, the computation of the Coeff-ideal only involves determining the Δ^i of the ideal which has previously been discussed and basic operations on ideals such as taking powers and sums.

---

[10]For simplicity, the row and column indices used inside the submatrices will be the ones of the corresponding rows resp. columns in the Jacobian matrix.

[11]E_j denotes the $j \times j$ unit matrix. As before, we use row and column indices corresponding to those of M for simplicity
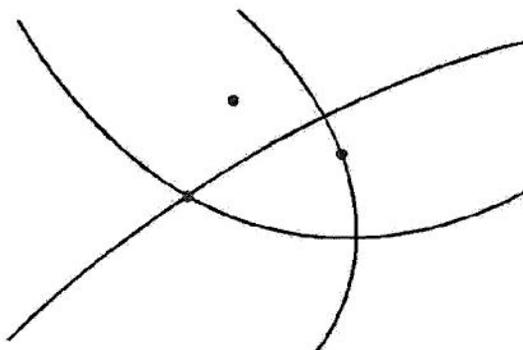
Figure 4: As an example for the problem of computing $\Delta(J)$, let us consider the situation illustrated in the above pictur: There are three minors whose determinant does not vanish (each one illustrated by one of the curves in the above picture) and $V(\Delta(J))$ consists of the three points. Then computing on the complement of just one of the minors will not provide all points of $V(\Delta(J))$, because each of the curves meets at least one point.

As H. Hauser already pointed out in his talk, such a hypersurface Z usually does not exist globally. In an implementation, the choice of the hypersurface involves passing to a suitable open covering such that on each open set U_j there is a hypersurface which can be used as Z for each point $w \in U_j$. The basic idea for finding such a covering is to consider $\Delta^{b'-1}(J)$ where J is the original ideal and b' is the maximal order of J. As the intersection of the singular loci of the generators of $\Delta^{b'-1}(J)$ is empty (b' is maximal order), it is possible to express 1 as a combination of the generators of the ideals of these singular loci and use the complements of those generators appearing with non-zero coefficients as the open covering[12]

The need to pass to an open covering can enlarge the number of charts significantly which slows down the subsequent steps of the resolution process due to duplicate calculations for points/subvarieties/centers appearing in more than one chart. The first idea to keep the number of open sets as low as possible is to recombine in the end, just as in the previous algorithm. Unfortunately, the auxilliary objects do depend on the chosen hypersurface, although the resulting value of the governing function at each point is independent of this choice. Therefore, we cannot just recombine directly as before; instead, we continue with the algorithm for finding the maximal locus of the governing function in each of the open sets and then (carefully) recombine those maximal loci.

---

[12] Of course, it is necessary to check that the two normal crossing conditions hold and, if necessary, pass to a different way of expressing 1 in terms of the generators of the singular loci.

4.2.4 Identification of Exceptional Divisors

The task here is to identify points resp. subvarieties which occur in more than one chart; in particular, we need to decide whether two given exceptional divisors living in two different charts actually belong to the same exceptional divisor (after glueing the charts). To this end, we move through the tree of charts arising during the resolution process, first blowing-down from the first chart to the one in which the history of the two charts in question branched, and then blowing-up again to the other chart with which we want to compare (cf. figure ??).
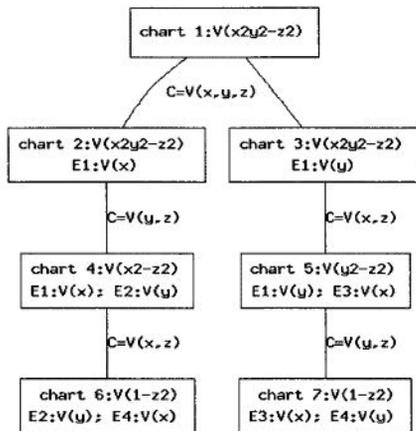

Figure 5: The tree of blow-ups in the resolution of the singularity $V(x^2y^2-z^2) \subset \mathbb{A}^3$. For simplicity of notation the variable names in all charts have been chosen to be x, y, z. To determine whether two exceptional divisors in two different charts actually belong to the same exceptional divisor, we need to move through the tree by first blowing down and then up again; for instance, the question, whether the divisors $V(x)$ in chart 6 and $V(y)$ in chart 7 belong to the same divisor, can only be answered by comparing the centers in charts 4 and 5. To this end, we have to move from chart 4 to chart one by blowing down twice and then proceed to chart 5 by blowing up twice.

As blow-ups are isomorphisms away from the center, this process of sucessively blowing-down and then blowing-up again does not cause any problems for points which do not lie on an exceptional divisor at all or only lie on exceptional divisors, which already exist in the chart at which the history of the considered charts branched. If, however, the point lies on an exceptional divisor which arises later, then blowing-down beyond the moment of birth of this divisor will inevitably lead to incorrect results, because this blow-up map is not an isomorphism. To avoid this problem, we need to represent the point on the exceptional divisor as the locus of intersection of the exceptional divisor with

34

an auxilliary variety which is not contained in the exceptional divisor. More formally speaking, we use the following simple fact from commutative algebra:

Let $I \subset K[x_1, \ldots, x_n]$ be a prime ideal, $J \subset K[x_1, \ldots, x_n]$ another ideal such that $I + J$ is equidimensional and $ht(I) = ht(I + J) - r$ for some integer $0 < r < n$. Then there exist polynomials $p_1, \ldots, p_r \in I + J$ and a polynomial $f \in K[x_1, \ldots, x_n]$ such that

$$\sqrt{I + J} = \sqrt{(I + (p_1, \ldots, p_r)) : f}.$$

In our situation, the ideal I is, of course, the ideal of the intersection of the exceptional divisors in which the point or subvariety $V(J)$ is contained. As any sufficiently general set of polynomials $p_1, \ldots, p_r \in J \setminus (I \cap J)$ leading to the correct height of $I + (p_1, \ldots, p_r)$ will do and as the only truely restricting condition on f is that it has to exclude all extra components of $I + (p_1, \ldots, p_r)$, we also have enough freedom of choice of the $p_1, \ldots, p_r, f$ to achieve that none of them is contained in any further exceptional divisor that might be in our way when blowing-down. Having solved the problem of identifying points which exist in more than one chart, we can now determine which exceptional divisor in one chart coincides with which one in another chart by simply comparing the centers leading to these exceptional divisors. To this end, we start at the root of the tree of charts of the resolution and work our way up to the final charts. The criteria for identifying the centers are quite simple: first of all, the centers can not be the same, if the the corresponding values of the governing function do not agree, secondly, the centers cannot be the same if the exceptional divisors in which they are contained are not the same and, in the last step, the remaining candidates are compared explicitly by mapping them through the resolution tree as described above.

At this point, we would like to repeat that computations in a computer algebra system are performed over $\mathbb{Q}$ not over the complex numbers although the reasoning often takes place over $\mathbb{C}$. This is particularly important to keep in mind for interpreting the results, e.g. if we need to determine the correct number of exceptional divisors arising during the resolution process. This will be an important issue in the subsequent section.

4.2.5 Intersection Matrix of Exceptional Curves

Given an embedded resolution of a surface singularity, stored as a tree of charts, we would like to pass to a non-embedded resolution by dropping unnecessary blow-ups at the end of the branches of the tree of charts. To this end, we compute the list of exceptional divisors by identifying them in the different charts as described in the previous section. Starting at the final charts, we then move backwards through the resolution tree and cancel those blowing ups which are not necessary for the non-embedded resolution (see illustration ?? for an example).
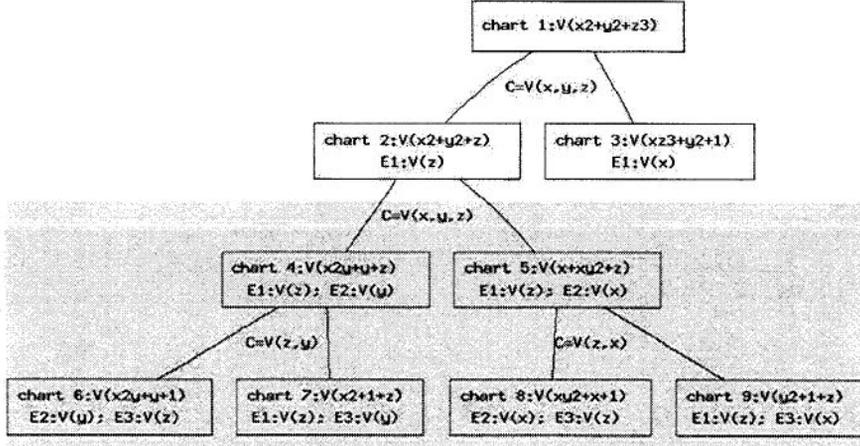
Figure 6: Tree of the embedded resolution process of an A_2 surface singularity. All charts which are marked by grey background arise from blow-ups which are only necessary in the embedded case, but not for a non-embedded resolution.

Then we consider the intersection of the remaining exceptional divisors of the embedded resolution with the strict transform to obtain the exceptional locus of the non-embedded resolution. We can easily decompose these intersection s into irreducible components over $\mathbb{Q}$,but these components may still be reducible over $\mathbb{C}$ and hence we need to achieve a descomposition over $\mathbb{C}$ to compute the intersection matrix of the exceptional divisors. The following theorem (cf. [?]) is the basis for the decomposition over $\mathbb{C}$:

Theorem 1 (Gao/Ruppert) *Let $f \in \mathbb{Q}[x,y]$ be irreducible of bidegree (m,n). Let $G = \{g \in \mathbb{Q}[x,y] | (m-1,n) \geq deg(g), \exists h \in \mathbb{Q}[x,y], \frac{\partial(g/f)}{\partial y} = \frac{\partial(h/f)}{\partial x}\}$. The vector space G has the following properties*

(i) *f is irreducible in $\mathbb{C}[x,y]$ if and only if $dim_{\mathbb{Q}}(G) = 1$.*

(ii) *$gG \subset \frac{\partial f}{\partial x}G \mod f$ for all $g \in G$.*

(iii) *Let $g_1, \ldots, g_a \in G$ be a basis and $g \in G \setminus \mathbb{Q}\frac{\partial f}{\partial x}, gg_i = \sum a_{ij}g_j\frac{\partial f}{\partial x} \mod f$. Let $\chi(t) = det(tE - (a_{ij}))$ be the characteristic polynomial. Then χ is irreducible in $\mathbb{Q}[t]$.*

(iv) *$f = \prod_{c \in \mathbb{C}, \chi(c)=0} gcd(f, g - c\frac{\partial f}{\partial x})$ is the decomposition of f into irreducible factors in $\mathbb{C}[x,y]$.*

We use this theorem for the decomposition of curves in $\mathbb{C}^n$ which are irreducible over $\mathbb{Q}$ by means of the following corollary:

Corollary 2 *Let $I \subset \mathbb{Q}[x_1, \ldots, x_n], ht(I) = 1$, be a prime ideal. Then there exists an irreducible polynomial $\chi(t) \in \mathbb{Q}[t]$ such that the complex zeros of $\chi(t) = 0$ correspond to the associated prime ideals of $I\mathbb{C}[x_1, \ldots, x_n]$.*

```
ring R=0,(x,y),dp;
poly p=x3-2y3;
getMinpoly(p);
```

```
[1]:
   poly p=t^3-2;
[2]:
   [1]:
      (-0.62996052494743658238360531+i*1.0911236359717214035600726)
   [2]:
      (-0.62996052494743658238360531-i*1.0911236359717214035600726)
   [3]:
      1.2599210498948731647672106121
[3]:
   3
```

If we factorize $x^3 + 2y^3$ over the field extension $\mathbb{Q}[t]/t^3 - 2$ we obtain two factors.

```
ring T=(0,t),(x,y),dp;
minpoly=t3-2;
factorize(x3-2y3);
```

```
[1]:
   _[1]=1
   _[2]=x2+(t)*x*y+(t2)*y^2
   _[3]=x+(-t)*y
[2]:
   1,1,1
```

To obtain a complete factorization we need a Galois extension which is of higher degree. Therefore the factorization takes more time and so do all further calculations in this field (Just consider the large coefficients of y in our very simple example!).

```
ring T=(0,t),(x,y),dp;
minpoly=t6+3t5+6t4+11t3+12t2-3t+1;
factorize(x3-2y3);
```

```
[1]:
   _[1]=1
   _[2]=x+(2/9t5+7/9t4+14/9t3+26/9t2+37/9t+2/9)*y
   _[3]=x+(1/9t5+2/9t4+4/9t3+4/9t2-1/9t-11/9)*y
```

37

```
_[4]=x+(-1/3t5-t4-2t3-10/3t2-4t+1)*y
```
[2]:
```
    1,1,1,1
```

To identify the $\mathbb{C}$-components of an exceptional divisor E (irreducible over $\mathbb{Q}$) in a chart, we, therefore, store E, $\chi(t)$ and the respective numerical root of $\chi(t)$. Given these data, we can then proceed in the same way as for the identification of the $\mathbb{Q}$-components in the previous section. As soon as the exceptional divisors in the different charts are identified, we can directly compute the intersection numbers $E_i.E_j$ for all $i \neq j$.

The computation of the self-intersection numbers E_i^2 is done by the following well-known method:
Let $\pi : X \longrightarrow Y$ be a resolution of the surface Y and $E_1, \ldots, E_s$ the exeptional divisors and let $h : Y \longrightarrow \mathbb{C}$ be a non-trivial linear form. Then $\pi^*(h).E_i = 0$. Now we can write

$$\pi^*(h) = \sum_{i=1}^{s} c_i E_i + H,$$

where H denotes the strict transform, and obtain the equations

$$0 = \pi^*(h)E_i = \sum_{i=1}^{s} c_j E_j.E_i + H.E_i \quad \forall 1 \leq i \leq s$$

which provide us with the desired self-intersection numbers.

References

[1] Arnold,V., Gusein-Zade,S., Varchenko,A.: *Singularities of Differentiable Maps I*, Birkhäuser (1985)

[2] Campillo,A.: *Algebroid Curves in Positive Characteristic*, Springer (1980)

[3] de Jong,T., Pfister,G.: *Local Analytic Geometry*, Vieweg, (2000)

[4] Ebeling,W.: Notes to the series of Talks entitled *Monodromy of Isolated Singularities* at this summer school

[5] Frühbis-Krüger,A.: *Construction of Moduli Spaces for Space Curve Singularities* JPAA 164 (2001), 165-178

[6] Frühbis-Krüger,A., Pfister,G.: *ome Applications of Resolution of Singularities from a Practical Point of View*, in Proceedings of Computational Commutative and Non-commutative Algebraic Geometry, Chisinau 2004 (2005), 104–117

[7] Gao,S.:*Factoring Multivariate Polynomials via Partial Differential Equations*, Math.Comp **72** (2003), 801–822

[8] Greuel,G.-M., Pfister,G.: *A* SINGULAR *Introduction to Commutative Algebra*, Springer (2002)

[9] Hauser,H.: Notes to the series of Talks entitled *The Proof of Resolution of Singularities in Characteristic Zero* at this summer school

[10] Labs,O.: *A Septic with 99 Real Nodes*, AG/0409348

[11] Looijenga,E.: *Isolated Singular Points of Complete Intersections*, Cambridge University Press, LNS 77 (1984)

[12] Schulze,M.: *A Normal Form Algorithm for the Brieskorn Lattice*, J.Symb.Comp.**38** (2004), 1207–1225

[13] Steenbrink,J.: Notes to the series of Talks entitled *Mixed Hodge Structures* at this summer school

[14] Wall,C.T.C.: *Singular Points of Plane Curves*, London Mathematical Society Student Texts 63 (2004)