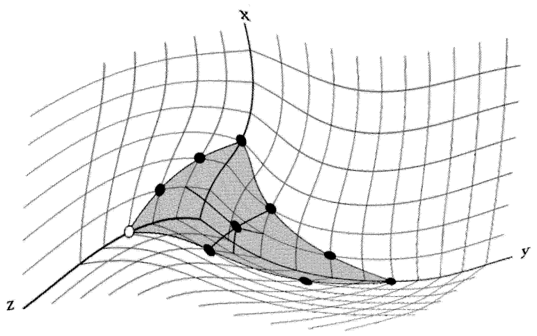
Klassifikation von Hyperflächensingularitäten

Kai Krüger
Betreuer: Prof. Dr. Gert-Martin Greuel
FB Mathematik
Universität Kaiserslautern



1997

Inhaltsverzeichnis

Einleitung

Diese Diplomarbeit gehört in das Gebiet der Singularitätentheorie. Es handelt sich um die Implementierung des Klassifizieres für Hyperflächensingularitäten nach Arnold [AGV] in dem Computeralgebrasystem Singular [Sin]. Durch die Entwicklung moderner Computeralgebrasysteme sind viele Berechnungen auf dem Computer einfach und oft wesentlich schneller und sicherer als per Hand durchführbar, wie z.B. Koordinatentransformationen, Berechnung von Dimension und Multiplizität oder von Syzygien. Dadurch ergibt sich die Möglichkeit algorithmisch formulierte theoretische Ergebnisse wie den Klassifizierer von Arnold auch praktisch zu implementieren. Der Klassifizierer von Arnold besteht aus 105 Sätzen und ist ähnlich wie im Bestimmungsbuch der Botanik aufgebaut. Er ist aber kein Algorithmus im strengen Sinne, da häufig nur die Existenz geeigneter Koordinatentransformationen gezeigt wird, ohne eine Methode zu ihrer Berechnung anzugeben. Die Herausforderung dieser Arbeit lag darin, Verfahren zu finden die es dem Computer ermöglichten, die zur Klassifikation notwendigen Koordinatentransformationen zu bestimmen. Da dem Computer jegliche menschliche Intuition fehlt, sind es oftmals nicht die besten, aber sie führen zum Ziel.

Im ersten Kapitel werden die im folgenden benötigten Begriffe und Aussagen kurz dargestellt. Die weiteren Kapitel stellen den Algorithmus und parallel dazu seine Implementierung vor. Aufgrund seiner Wichtigkeit für die Klassifikation ist dem Spaltungslemma und seiner Implementierung das gesamte zweite Kapitel vorbehalten. Im dritten Kapitel folgt dann der eigentliche Algorithmus zur Klassifikation. Er zerfällt in zwei Teile: den Klassifzierer für Singularitäten vom Corang 2 und den für Corang 3, d.h. der Corang der Hesse-Matrix ist im einen Fall 2, in dem anderen 3. Während im Fall "Corang 2" eine vollständige Implementierung möglich war, ist diese in Corang 3 nicht vollständig. So war es nicht in jedem Fall möglich, ohne Übergang zu einer Körperereiterung oder ohne Polynomfaktorisierung einerseits und ohne menschliche Intuition andererseits den gewichteten Jet auf Normalform zu bringen. Außerdem fehlt im Schritt 90 bei Arnold ein Fall. Es ist beabsichtigt, diese Lücken in Corang 3 in naher Zukunft zu schließen.

Bei der Implementierung des Klassifizierers zeigte sich darüberhinaus, daß das Hilbertpolynom der Milnoralgebra eine sehr feine Invariante ist, die es in vielen Fällen erlaubt, den Typ der Singularität bereits anzugeben. In einzelnen Fällen sind sogar Aussagen über Moduli möglich. Auf dieser Beobachtung beruht der zweite Klassifikationsansatz, der in Anhang A beschrieben wird. Es ist beabsichtigt, auch diese zweite Variante des Klassifizierers weiterzuentwickeln.

Ich möchte Herrn Professor Gert-Martin Greuel für die Betreuung der Diplomarbeit danken, ebenso wie Herrn Professor Duco van Straten für viele hilfreiche und anspornende Diskussionen. Ein besonderer Dank geht auch an die Entwickler von Singular, denn ohne dieses Computer-Algebra System wäre allein schon die Aufgabenstellung in dieser Form nicht möglich gewesen, und an Hans Schönemann, der mir oft als lebendes Singular-Handbuch Rede und Antwort stand und für den Klassifizierer notwendige Funktionen des Singularkerns auch für den Nutzer zugänglich machte.

Kapitel 1

Allgemeine Grundlagen

1.1 Keime und Jets

Definition 1.1 *[Mar, S. 1]*

Ein **Keim** *einer Abbildung $f : \mathbb{C}^n \to \mathbb{C}^k$ in einem Punkt $p \in \mathbb{C}^n$ ist eine Äquivalenzklasse von Abbildungen $g : U_g \to \mathbb{C}^k$, wobei U_g eine Umgebung von p ist, bezüglich folgender Äquivalenzrelation:*
Seien $f_1 : U_1 \to \mathbb{C}^k$ und $f_2 : U_2 \to \mathbb{C}^k$ zwei Funktionen, dann gilt:
f_1 ist äquivalent zu f_2 (in Zeichen $f_1 \sim f_2$) $:\Leftrightarrow$ Es existiert eine Umgebung $U \subseteq U_1 \cap U_2$ von p mit $f_1 |_U = f_2 |_U$.

Wir bezeichnen mit $\mathcal{O}_{(\mathbb{C}^n,p)}$ die Algebra aller holomorphen Keime $f : \mathbb{C}^n \to \mathbb{C}$ in $p \in \mathbb{C}^n$.

Definition 1.2

Sei f ein holomorpher Keim in p. Dann wird mit

$$J_f := \left(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n}\right) \subset \mathcal{O}_{(\mathbb{C}^n,p)}$$

das **Jacobi Ideal** *von f bezeichnet, die $\mathbb{C}$-Algebra $\mathcal{O}_{(\mathbb{C}^n,p)}/J_f$ heißt* **Milnor-Algebra**

Beispiel 1.3

Für $f = x^2y + y^k$ ist $J_f = (xy, x^2 + ky^{k-1})$

Definition 1.4 *[AGV, 5.1]*

Die **Milnor Zahl** *eines holomorphen Keimes f in einem Punkt p ist gerade die Dimension der Milnor-Algebra:*

$$\mu_p(f) := dim_{\mathbb{C}}(\mathcal{O}_{(\mathbb{C}^n,p)}/(J_{f,p}))$$

Statt $\mu_0(f)$ schreiben wir auch $\mu(f)$.

Definition 1.5

Ein Keim f hat einen **isolierten kritischen Punkt** *(oder eine* **isolierte Singularität***) in p, wenn $\mu_p(f) < \infty$ gilt.*

Beispiel 1.6
$x^2 + y^k$ hat 0 in $(\mathbb{C}^2, 0)$ als isolierten kritischen Punkt mit Milnorzahl $k-1$.
x^2 hat in $(\mathbb{C}^2, 0)$ keinen isolierten kritischen Punkt.

In dieser Arbeit werden wie bei Arnold holomorphe Funktionskeime bis auf Rechtsäquivalenz klassifiziert. Dabei heißen $f, g \in \mathcal{O}_{(\mathbb{C}^n, p)}$ **rechts-äquivalent** (oder einfach **äquivalent**), in Zeichen $f \sim_R g$ (oder einfach $f \sim g$), falls es einen analytischen Automorphismus φ von $\mathcal{O}_{(\mathbb{C}^n, p)}$ gibt, sodaß $\varphi(f) = g$ gilt.

Definition 1.7 *[Mar, S. 175]*
Ist $f \in \mathcal{O}_{(\mathbb{C}^n, p)}$, so bezeichne $j_k f$ den **k-jet von** *f, d.h. das Taylorpolynom von f mit Entwicklungspunkt p bis zum Grad k (inklusive).*
f heißt **endlich bestimmt***, falls es ein $k \in \mathbb{N}$ gibt, so daß für jedes $g \in \mathcal{O}_{(\mathbb{C}^n, p)}$ mit $j^k f = j^k g$ gilt: $f \sim_R g$.*
Man sagt auch f ist **k-bestimmt**.

Satz 1.8 *[Mar, S.24]*
Sei $f \in \mathcal{O}_{(\mathbb{C}^n, p)}$. Falls es ein $k \in \mathbb{N}$ gibt mit $\mathcal{M}^k \subseteq \mathcal{M} \cdot Jf$, so ist f k-bestimmt. Hierbei bezeichne $\mathcal{M}$ das mazimale Ideal $(x_1 - p_1, \ldots, x_n - p_n) \subset \mathcal{O}_{(\mathbb{C}^n, p)}$.

Corollar 1.9
Sei $f \in \mathcal{O}_{(\mathbb{C}^n, p)}$ und $\mu = \mu_p(f) < \infty$. Dann ist f $\mu + 1$-bestimmt.

Beweis

$$\begin{gathered} dim_{\mathbb{C}}(\mathcal{M}/J_f) = \mu - 1 \\ \Rightarrow (\mathcal{M}/J_f)^\mu = (\mathcal{M}^\mu + J_f)/J_f = 0 \\ \Rightarrow \mathcal{M}^\mu \subset J_f \\ \Rightarrow \mathcal{M}^{\mu+1} \subset \mathcal{M} \cdot J_f \end{gathered}$$

■

Bemerkung 1.10
Die Milnorzahl μ, und damit eine Abschätzung für die Bestimmtheit, läßt sich mit Singular schnell berechnen, aber in der Regel ist diese Abschätzung sehr schlecht.
Eine bessere Abschätzung liefert die in Singular implementierte Funktion `system("HC")`*, wobei “HC” für “highest Corner” (vgl[GP]) steht.*

Beispiel 1.11

Sei $f = (y - x^2)^2 + y^4 = y^2 - 2x^2y + x^4 + y^4$
Behauptung : f *ist 8-bestimmt, die Milnorzahl ist 7.*
$\frac{\partial f}{\partial x} = -4xy + 4x^3$
$\frac{\partial f}{\partial y} = -2x^2 + 2y + 4y^3$

Damit gilt:

$$\mathcal{M} \cdot Jf \;=\; \mathcal{M}\cdot < -4xy + 4x^3, -2x^2 + 2y + 4y^3 >$$

$$= < \underbrace{-4x^2y + 4x^4}_{g_1}, \underbrace{-2x^3 + 2xy + 4xy^3}_{g_2}, \underbrace{-4xy^2 + 4x^3y}_{g_3}, \underbrace{-2x^2y + 2y^2 + 4y^4}_{g_4} >$$

$\Rightarrow g_1 + x \cdot g_2 = 2x^2y^3 \in \mathcal{M} \cdot Jf$
$x^2y^3 + xy \cdot g_3 = x^4y^2 \in \mathcal{M} \cdot Jf$
$x^2y^3 + y^2 \cdot g_4 = (1 + 2y^2)y^4 \in \mathcal{M} \cdot Jf$
$x^2 \cdot g_2 + x \cdot g_4 - 2x \cdot y^4 - 2x \cdot x^2y^3 = -x^5 + xy^2 \in \mathcal{M} \cdot Jf$
$x^3 \cdot (-x^5 + xy^2) - x^4y^2 = x^8 \in \mathcal{M} \cdot Jf$
$xy \cdot (-x^5 + xy^2) - x^2y^3 = x^6y \in \mathcal{M} \cdot Jf$

$$\Rightarrow \mathcal{M}^8 \subseteq < x^8, x^6y, x^4y^2, x^2y^3, y^4 > \subseteq \mathcal{M}Jf$$

Nach dem vorherigen Satz ist f *somit 8-bestimmt.*
In Singular liefert `system(''HC'')` *den Wert 8.*

Definition 1.12 *[AGV, 12.1]*
Seien $w_1, \ldots, w_n$ *positive ganze Zahlen.* $f \in \mathbb{C}\{x_1, \ldots, x_n\}$ *heißt* **quasihomogen** *(oder* **gewichtet homogen***) vom* **w-Grad** *(oder einfach* **Grad***)* d*, falls* $f(\lambda^{w_1}x_1, \ldots, \lambda^{w_n}x_n) = \lambda^d f(x_1, \ldots, x_n)$ *für alle* $\lambda \in \mathbb{C}$ *gilt.*
Der **(w-)Grad** *eines Monoms* $x_1^{i_1} \cdot \ldots \cdot x_n^{i_n}$ *ist* $< w, i > = w_1i_1 + \ldots + w_ni_n$.

Beispiel 1.13
$J_{2,0} : 7x^3 + 2x^2y^2 - 5y^6$ *ist quasihomogen mit den Gewichten (2,1) vom w-Grad 6.*
$J_{2,k} : x^3 + x^2y^2 + y^{6+k}$ *mit* $k > 0$ *ist nicht quasihomogen für irgendwelche Gewichte* w_1, w_2.
In der Tat ist $J_{2,k}$, $k > 0$, *für keine lokalen holomorphen Koordinaten quasihomogen, da* $f \notin J_f$ *für* $f = x^3 + x^2y^2 + y^{6+k}$*. Das Singular-Kommando* `NF(`f`,std(jacob(`f`)))` *liefert (für verschiedene Werte von k)* c_ky^{6+k} *wobei* $c_k \neq 0$ *ist.*

Definition 1.14 *[AGV, 12.1]*

*Ein Polynom (bzw. eine Potenzreihe) $\neq 0$ hat (w-)***Ordnung** *d, wenn alle seine Monome mindestens den (w-)Grad d haben und es mindestens ein Monom gibt, das den (w-)Grad d besitzt, die 0 hat die Ordnung ∞.*

Die Polynome (bzw. Potenzreihen) der (w-)Ordnung $\geq d$ bilden einen $\mathbb{C}$-Vektorraum A_d. Es gilt $A_{d'} \subset A_d$ für $d' > d$.

Die (w-)Ordnung eines Produktes ist gerade die Summe der Ordnungen, und daher ist A_d ein Ideal in $A = \mathbb{C}[x_1, \ldots, x_n]$ (bzw. $A = \mathbb{C}\{x_1, \ldots, x_n\}$).

Notation:

Im folgenden bezeichne A stets $\mathbb{C}[x_1, \ldots, x_n]$ (bzw. $\mathbb{C}\{x_1, \ldots, x_n\}$) und $\mathcal{M}$ das maximale Ideal $(x_1, \ldots, x_n)$. Positive ganze Zahlen $w_1, \ldots, w_n$ seien fest gewählt.

Definition 1.15 *[AGV, 12.1]*

Die Quotientenalgebra A/A_d heißt die Algebra der **d-(quasi-)jets**.
Ein Element von A/A_d nennt man auch **gewichteten Jet**.

Lemma 1.16 *[AGV, 12.4]*

Sei $f \in \mathbb{C}[x_1, \ldots, x_n]$ quasihomogen vom w-Grad d und $\varepsilon \in \mathbb{C}\backslash\{0\}$. Dann gilt:

$$f \approx \varepsilon f,$$

wobei $\approx$ quasihomogene Äquivalenz bedeutet.

Beweis

Die quasihomogene Koordinatentransformation $x_i \mapsto \varepsilon^{\frac{w_i}{d}} x_i$ überführt f in εf, da $f(\varepsilon^{\frac{w_1}{d}} x_1, \ldots, \varepsilon^{\frac{w_n}{d}} x_n) = \varepsilon f(x_1, \ldots, x_n)$ gilt.
$\blacksquare$

Definition 1.17 *[Eis, S.360]*

Seien $I, J \subset A$ Ideale. Das Ideal

$$I : J := \{f \in A | f \cdot J \subset I\}$$

heißt der **Idealquotient von** *I und J.*
Das Ideal $I : J^\infty = \bigcup_{k \geq 0} I : J^k$ heißt die **Sättigung** *(oder* **Saturierung***) von I* **bzgl** *J.*
Das Ideal

$$\bar{I} := I : \mathcal{M}^\infty = \{f \in A \mid (x_1, \ldots, x_n)^k \cdot f \subset I \text{ für ein } k > 0\}$$

heißt die **Sättigung** *von I.*

1.2 Invarianten

Definition 1.18 *[AGV, 11.1]*
Der **Corang** *eines Keimes* $f \in \mathcal{O}_{(\mathbb{C}^n,p)}$ *in einem kritischen Punkt* p *ist der Corang der Hesse-Matrix* $H_f(p)$ *von* f *in* p,

$$H_f(p) = \begin{pmatrix} \frac{\partial^2 f(p)}{\partial x_1^2} & \cdots & \frac{\partial^2 f(p)}{\partial x_1 x_n} \\ \vdots & & \vdots \\ \frac{\partial^2 f(p)}{\partial x_n x_1} & \cdots & \frac{\partial^2 f(p)}{\partial x_n x_n} \end{pmatrix}.$$

Er wird mit $corang(f,p)$ *bezeichnet, wobei* $corang(f)$ *für* $corang(f,0)$ *steht.*

Beispiel 1.19

Für $f = y^2 + z^2 + x^6$ *ist* $H_f = \begin{pmatrix} 30x^4 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix}$.

Dann hat $H_f(0) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix}$ *den Rang 2, also Corang 1.*

Definition und Satz 1.20 *[Mat, S.95ff]*
$A = \mathbb{C}\{x_1, \ldots, x_n\}/I$. *Sei* $I \subset \mathbb{C}\{x_1, \ldots, x_n\}$ *ein Ideal und* $\mathcal{M}$ *das maximale Ideal von* A.
Setze $\psi(t) := dim_{\mathbb{C}}(A/\mathcal{M}^t)$. *Für genügend große* t *stimmt* $\psi(t)$ *mit einem Polynom* $P_A(t) \in \mathbb{Q}[t]$ *überein. Dieses wird als* **Hilbert-Polynom** *(oder* **Hilbert-Samuel-Polynom***) von* A *bezeichnet.*

Definition 1.21 *[Mat, S.98]*
Die **Dimension** *von* A *ist gegeben durch*

$$dim\ A = deg\ P_A.$$

Definition 1.22 *[Mat, S.107f]*
Die **Multiplizität** *von* A *ist gegeben durch* $(dimA)!$ *mal den Leitkoeffizienten von* P_A.

Kapitel 2

Spaltungs-Lemma

Im folgenden bezeichne $\sim$ stets rechts-äquivalent.

Definition 2.1 *[AGV, S.187]*
Zwei holomorphe Funktionskeime f und g heißen **stabil äquivalent**, *wenn sie nach Addieren einer nicht ausgearteten quadratische Form rechts-äquivalent sind, d.h.:*

$$f(x_1, \ldots, x_k) + x_{k+1}^2 + \ldots + x_n^2 \sim g(y_1, \ldots, y_l) + y_{l+1}^2 + \ldots + y_n^2$$

Satz 2.2 *[AGV, S.187](Spaltungslemma)*
Eine holomorpher Funktionskeim $f \in \mathcal{O}_{(\mathbb{C}^n,p)}$ mit kritischem Punkt p und Corang k ist äquivalent zu dem Funktionskeim $g(x_1, \ldots, x_k) + x_{k+1}^2 + \ldots + x_n^2$, d.h.
ist $f \in \mathcal{O}_{(\mathbb{C}^n,p)}$ mit $f \in \mathcal{M}^2$, $k = corang(f)$, dann gilt: $f(x_1, \ldots, x_n) \sim g(x_1, \ldots, x_k) + \sum\limits_{i=k+1}^{n} x_i^2$, mit $g \in \mathcal{M}^3$

Beispiel 2.3

$$f = 2x^2 + 4xy + 10y^2 - 6xz - 2yz + 5z^2 + y^4$$

$$= (x + 3y - z)^2 + (y - x + 2z)^2 + y^6 \sim x^6 + y^2 + z^2$$

mit $g(x) = x^6$

Definition 2.4
Die Funktion $g \in \mathcal{M}^3$ im vorigen Satz ist stabil äquivalent zu f und wird als **Restsingularität** *bezeichnet.*

Der Algorithmus des Spaltungslemmas, der in dieser Arbeit angegeben wird, basiert auf dem von F.Tari [Tar], der in vielen Fälle funktioniert, aber im Allgemeinen falsch ist. Dieser sieht wie folgt aus:

Algorithmus (Tari)
Input: $f \in \mathbb{C}\{x_1, \ldots, x_n\}$
Output: $g \in \mathbb{C}\{x_1, \ldots, x_k\}$,$k$ mit $k = Corang(f)$ und $f \sim g + \sum\limits_{i=k+1}^{n} x_i^2$

1. Sei S eine leere Indexmenge
2. Prüfe, ob f in 0 singulär ist;
   (a) Nein: return f, n
   (b) Ja: Weiter.
3. Setze $i := 1$.
4. Prüfe, ob x_i in $j_2 f$ vorkommt (wegen 2. kommen in f keine linearen Terme vor).
   (a) Nein:
   $\Rightarrow S = S \cup \{i\}$.
   (b) Ja:
      i. Schreibe $f = a{x_i}^2 + R(x_1, \ldots, x_n)$.
      ii. Ist $a = 0$, so wähle ein $j > i$, so daß der Koeffizient von $x_i x_j$ ungleich Null ist, und führe die Koordinatentransformation
      $$\begin{cases} x_l \mapsto x_l & , \; l \neq j \\ x_j \mapsto x_j + x_i \end{cases}$$
      durch.
      iii. Schreibe $f = a{x_i}^2 + P(x_{i+1}, \ldots, x_n)x_i + R(x_1, \ldots, x_n)$.
      iv. Definiere die Koordinatentransformation
      $$\varphi : \begin{cases} x_i \mapsto x_i - \frac{P(x_{i+1},\ldots,x_n)}{2a} \\ x_j \mapsto x_j \end{cases}$$
      um den Term Px_i zu eliminieren.
      v. Setze $f(x_1, \ldots, x_n) := (f \circ \varphi)(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n) + {x_i}^2$.
5. Erhöhe i um 1 und gehe zu Schritt 4.

6. return f, $\#S$

Bei der Implementierung des Algorithmus sind in Schritt *4(b)(iii)* und *4(b)(v)* zwei Fehler aufgefallen. Wie diese aussehen und wie man sie behebt, wird im Folgenden an Hand von zwei Beispielen illustriert.

Beispiel 2.5

Wende obigen Algorithmus auf $f = (y - x^2)^2 + y^4 - x^4 = y^2 - 2x^2y + y^4$ *an.*
Starte mit $S = \{\}$.
Betrachte zunächst x.
x *kommt in* $j_2(f)$ *nicht vor, also* $S = S \cup \{1\}$
Nun ist y *an der Reihe.* $f = ay^2 + Py + R$ *mit* $P = 0$ *und* $R = -2x^2y + y^4$
Damit ist φ *gerade die Identität, und es folgt* $f(x, y) := (f \circ \varphi)(x, y) = y^2 - 2x^2y + y^4$.
Ab jetzt können alle Terme der Restsingularität in y *ignoriert werden:* $f(x, y) \sim g(x) + y^2$ *mit Restsingularität* $g(x) = 0$.
Unser f *war jedoch eine* A_3*-Singularität, d.h* $g(x) \sim x^4$.

Dieses Beispiel zeigt, daß man im Algorithmus folgende Korrektur vornehmen muß:
Der Schritt 2(b)(iii) wird:
Schreibe $f = a{x_i}^2 + P(x_1, \ldots, \hat{x}_i, \ldots, x_n)x_i + R(x_1, \ldots, x_n)$

Somit erhalten wir im Beispiel
2(b)(iii): $f = ay^2 + Py + R$ mit $P = -2x^2$ und $R = y^4$
2(b)(iv): $\varphi : \begin{cases} x \mapsto x \\ y \mapsto y + x^2 \end{cases}$
2(b)(v): $f(x, y) \sim (f \circ \varphi)(x, 0) + y^2 = y^2 - x^4 + x^8$

Terme in y kommen in der Restsingularität nicht mehr vor. f ist also wie erwartet eine A_3-Singularität.

Beispiel 2.6

Wende die neue Form des Algorithmus auf $f = (x - y^2)^2 + x^4 - y^8 = x^2 - 2xy^2 + x^4 + y^4 - y^8$ *an.*
$S = \{\}$
Betrachte zunächst x.
x *kommt in* $j_2(f)$ *vor.*
$f = ax^2 + Px + R$ *mit* $P = -2y^2$ *und* $R = x^4 + y^4 - y^8$
Damit ist $\varphi : \begin{cases} x \mapsto x + y^2 \\ y \mapsto y \end{cases}$,
und es folgt $f(x, y) := (f \circ \varphi)(x, y) \sim x^2 + x^4 + 4x^3y^2 + 6x^2y^4 + 4xy^6$.

Ab jetzt können alle Terme der Restsingularität in x ignoriert werden: $f(x,y) \sim x^2 + g(y)$ mit Restsingularität $g(y) = 0$.
Unser f war jedoch eine A_{11}-Singularität, d.h $g(y) = y^{12}$.

Das Beispiel zeigt, daß die Terme in x keineswegs irrelevant sind. Wir haben dabei aber auch gesehen, daß der Term in xy^2 zu einem Term xy^6 wurde. Die Idee ist nun, den Algorithmus wiederholt auf dieses neue f bzgl. der gleichen Koordinate x anzuwenden, um damit zu erreichen, daß der Term $xP(y)$ über die k-Bestimmtheitsgrenze von f rutscht.

Betrachten wir noch einmal das letzte Beispiel.
Die Bestimmtheit von f ist 12, da die Milnorzahl 12 ist.
Im korrigierten Algorithmus liefert die Abbildung $\varphi_1 : \begin{cases} x \mapsto x - 2y^6 \\ y \mapsto y \end{cases}$ folgendes Ergebnis:

$j_{12}(f \circ \varphi \circ \varphi_1) = x^2 + x^4 + 4x^3y^2 + 6x^2y^4 - 8x^3y^6 - 24x^2y^8 \underline{-24xy^{10}}$
Im nächsten Schritt:
$\varphi_2 : \begin{cases} x \mapsto x + 12y^{10} \\ y \mapsto y \end{cases}$
$j_{12}(f \circ \varphi \circ \varphi_1 \circ \varphi_2) = x^2 + x^4 + 4x^3y^2 + 6x^2y^4 - 8x^3y^6 - 24x^2y^8 - 4y^{12}$
Nun kommt tatsächlich kein Term der Form $xP(y)$ mehr vor. Es können alle Terme der Restsingularität in x ignoriert werden, denn:
Schreibt man $f = x^2Q(x,y) + R(y)$, liefert eine Substitution $x_1 = x\sqrt{Q(\underline{x})}$ das gewüschte Quadrat.
Wir erhalten: $f \sim x^2 - 4y^{12}$, eine A_{12}-Singularität.

Damit ergibt sich also folgender Algorithmus:

Algorithmus 1 (Spaltungslemma)

Input: $f \in \mathbb{C}[x_1, \ldots, x_n]$, k so daß f k-bestimmt ist.
Output: $g \in \mathbb{C}[x_1, \ldots, x_c]$, $c \in \mathbb{N}$,
mit $c = Corang(f)$ und $f \sim g + \sum\limits_{i=c+1}^{n} x_i^2$

1. Sei S eine leere Indexmenge, $i = 0$;
2. Prüfe, ob f in 0 singulär ist;
   (a) Nein: return f, n
   (b) Ja: Weiter.

3. Setze $i := i + 1$. Notation: $\underline{x} = (x_1, \ldots, \hat{x}_i, \ldots, x_n)$.

4. Prüfe, ob x_i in $j_2 f$ vorkommt (wegen 2. kommen in f keine linearen Terme vor).

   (a) Nein:
   $\Rightarrow S = S \cup \{i\}$.

   (b) Ja:
   Schreibe $f = Q(\underline{x}, x_i)x_i^2 + P(\underline{x})x_i + R(\underline{x})$.

      i. $Q(0,0) = 0$:
      Für $j > i$ suche eine Koordinatentransformation

      $$\psi : \begin{cases} x_l \mapsto x_l \ , & l \neq j \\ x_j \mapsto x_j + x_i, & \end{cases}$$

      so daß $\psi(Q(0,0)) \neq 0$ (d.h $\psi(Q(\underline{x}, x_i))$ Einheit) .

      ii. $Q(0,0) \neq 0$:
      Definiere die Koordinatentransformation

      $$\varphi : \begin{cases} x_i \mapsto x_i - \frac{P(\underline{x})}{2Q(0,0)} \\ x_j \mapsto x_j \end{cases}$$

      iii. Ersetze $f := f \circ \varphi$ und erhalte $f = Q'(\underline{x}, x_i)x_i^2 + P'(\underline{x})x_i + R'(\underline{x})$ (damit erreicht man Ordnung$(P'(\underline{x})) >$ Ordnung $(P(\underline{x}))$.

      iv. Ersetze $f := j_k f$ und erhalte $f = Q'(\underline{x}, x_i)x_i^2 + P'(\underline{x})x_i + R'(\underline{x})$.

      v. Prüfe $P'(\underline{x}) = 0$
      (nach endlich vielen Schritten ist wegen iii. und iv. $P'(\underline{x}) = 0$.)

         A. Nein: Gehe *4.b.*

         B. Ja: Weiter

5. Falls $i < n$ gehe zu *3.*

6. Wir haben nun $f \sim g(x_{i \in S}) + \sum\limits_{i \notin S} x_i^2 \cdot Q_i(\underline{x})$ mit $g \in \mathcal{M}^3$ und $Q_i(\underline{0}) \neq 0$.
   Definiere die Koordinatentransformation

   $$\varphi : \begin{cases} x_i \mapsto x_i & , \quad i \in S \\ x_i \mapsto 0 & , \quad i \notin S \end{cases}$$

   und setze $g := f \circ \varphi$.
   (Damit sind wir fertig, denn für den lokalen Ring $\mathbb{C}\{x_1, \ldots, \mathbf{x}_n\}$ läßt sich folgende Koordinatentransformation definieren:

   $$\varphi : \begin{cases} x_i \mapsto x_i \cdot \sqrt{Q_i'(\underline{x})} \ =: \ x_i' & , \quad i \notin S \\ x_i \mapsto x_i & , \quad i \in S \end{cases}$$

und damit gilt: $f \sim g(x_{i\in S}) + \sum\limits_{i\notin S}(x'_i)^2$).

7. return g, $\#S$

Anmerkungen:

- Die Mächtigkeit der Indexmenge S ist gerade der Corang(f).
- Die gesamte Implementierung arbeitet in einem Ring mit den Variablen $x(1), \ldots, x(n)$.
- In der Praxis hat sich gezeigt, daß es effizienter ist, zuerst die Koordinaten zu bearbeiten, die am seltensten im Polynom f vorkommen und erst ganz zum Schluß diejenigen, die am häufigsten in f vorkommen. Die Bearbeitungsreihenfolge ist als Intvektor in der Variablen $RFlg[]$ abgespeichert. (Je häufiger eine Variable vorkommt, desto mehr bläht sich nämlich f nach einer Koordinatentransformation in dieser Variablen auf, und damit sind eine längere Laufzeit und ein höherer Speicherbedarf verbunden.)
- Die Funktion BasicInvariants() wird vor Aufruf des Algorithmums aufgerufen. Sie liefert die in der Implementierung benötigten Invarianten wie Corang(f), Milnorzahl und k-Bestimmtheit von f.
  Diese werden in Singular wie folgt berechnet:

  ```
  ideal Jf= jacob(f);
  ideal Jfs= std(Jf);
  int hc= system("HC") + 1;
  int mu= vdim(Jfs);
  int corang= dim(std(jacob(jet(f, 2)))); [1]
  int k= min(hc,mu) + 1;
  ```

  Hierbei ist: (f) das Ideal $(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n})$.
  Das Kommando `std` berechnet eine Standardbasis im lokalen Ring.

  Bei der k-Bestimmtheit wird das Minimum der “Höchsten Ecke” und der Milnorzahl, jeweils um eins erhöht, zurückgeliefert.
  Die Milnorzahl ist nur eine obere Schranke für die k-Bestimmtheit. In einigen Fällen ist weit größer als k. HC ist die höchste Ecke, die bei der

[1] Weitere Möglichkeit zur Berechnung:
corank = `nvars(basering) - size(module(transpose(bareiss(jacob(jacob(jet(f,2)))))));`

letzten Standardbasenberechnung als Abbruchschranke diente. Es gilt also $HC \leq \mu$.
In der verwendeten Singularversion wird HC auf einen Wert von 32000 initialisiert und erhält erst während der Standardbasenberechnung seinen Wert. Ist das Ideal jedoch bereits monomial so findet keine Berechnung statt und HC hat noch den Initialisierungswert.

- Eine weitere Optimierung ist für den Fall eingebaut, daß der Corang von f um 1 niedriger ist als die Zahl der Ringvariablen.
Dann läßt sich f als Summe einer linearen Form mit einem Rest in einer Veränderlichen weniger schreiben, d.h. $f = (\sum_{j=1}^{n} a_j x_j)^2 + g(x_1, \ldots, \hat{x}_i, \ldots, x_n)$ mit $a_i \neq 0$ und $g \in \mathcal{M}^3$.
Suche ein $y = x_i$ so, daß $\frac{\partial(j_2 f)}{\partial x_i} \neq 0$. Berechne dann die Syzygie (r_1, r_2) von $j_2 f$ mit $\frac{\partial(j_2 f)}{\partial x_i}$. Wir erhalten (r_1 vom Grad 0, r_2 vom Grad 1):

$$
\begin{gathered}
(r_1, r_2) \cdot (j_2 f, \frac{\partial(j_2 f)}{\partial x_i})^T = 0 \\
r_1(\sum a_j x_j)^2 + 2a_i r_2(\sum a_j x_j) = 0 \\
\Leftrightarrow r_1(\sum a_j x_j) = -2a_i r_2 \\
\Leftrightarrow \sum a_j x_j = -2a_i \frac{r_2}{r_1}
\end{gathered}
$$

Hieraus erhält man dann die Koordinatentransformation

$$
\begin{cases} x_i \mapsto 2\frac{r_2}{r_1} & \\ x_j \mapsto x_j & , j \neq i, \end{cases}
$$

die das Gewünschte liefert.
Es hat den Vorteil, daß man das Resultat in einem Schritt erreicht. Führt man den normalen Algorithmus durch, so muß jede Koordinate getestet werden und für jede von ihnen mindestens eine Koordinatentransformation ermittelt werden. Der Zeitverlust liegt also auf der Hand.

Kapitel 3

Der Klassifizierer nach Arnold

Das folgende Kapitel beschreibt, wie der Klassifikationsalgorithmus von Arnold [Arn] in Singular implementiert wurde, und parallel dazu die Theorie aus dem gleichen Werk. Zur Erleichterung des Vergleiches von Algorithmus und Programm sind die internen Funktionen der Implementierung mit Funktion$< n >$ benannt. n steht hierbei für den jeweiligen Schritt im Algorithmus bei Arnold.

Die Koordinatentransformationen, von denen im Klassifizierer von Arnold gezeigt wird, daß sie existieren, werden im Algorithmus nur soweit durchgeführt, wie sie zur Bestimmung des Typs der Singularität innerhalb einer Klasse notwendig sind. Die Koordinatentransformationen, die zur Bestimmung der Normalform (d.h. zu der Bestimmung der genauen Parameter eines Types) nötig wären, werden nicht durchgeführt.
Dabei werden die Begriffe “Klasse”, “Typ” und “Normalform” wie im folgenden Beispiel verwendet. Die Singulartität

$$f(x,y) = \varepsilon x^3 + \beta x^2 y^3 + xy^7$$

mit $\varepsilon \neq 0$, $\beta \neq 0$ gehört zur Klasse J und hat den Typ J$[3,2]$. Die Normalform von f hat nach Arnold die Gestalt: $x^3 + x^2y^3 + ay^{11}$, $a = a_0 + a_1 y$, $a_0 \neq 0$.
Die Koordinatentransformation, die zu dieser Normalform führen würde, wird nicht mehr bestimmt, und damit wird der Paramter a auch nicht berechnet.

Im gesamten Klassifizierer kommt es darauf an, ob eine Monom in einem Jet vorkommt oder nicht. Kommt es mit nicht verschwindenden Koeffizienten ε vor, so wird die Koordinatentransformation, die ε auf 1 normiert, nicht durchgeführt. Die Notation:

$$\varepsilon x^i y^j z^k$$

bedeutet, daß $\varepsilon \neq 0$ ist. Ebenso wird die quasihomogene Koordinatentransformation, die ein quasihomogens εf in f überführt (Lemma 1.15), nur durchgeführt falls nötig.

3.1 Einteilung nach corang (f)

Satz 3.1

Sei $f \in \mathcal{O}_{(\mathbb{C}^n,0)}$, mit $f \in \mathcal{M}^2$, $f \sim_R \sum_{i=1}^{p} x_i^2 + g(x_{p+1}, \ldots, x_n)$ mit $p = RangHf(0)$ für geeignetes $g \in \mathcal{M}^3$.
Um f zu klassifizieren genügt es, die Klasse der Restsingularität g zu bestimmen.

Damit ist corang(f) unser erstes Kriterium zur Klassifikation, und wir erhalten folgende Tabelle:

corang(f)		Serie	Bemerkung	Schritt
0	$f \approx \sum_{i=1}^{n} x_i^2$	A_1		2
1	$f \approx \sum_{i=1}^{n-1} x_i^2 + x_n{}^{k+1}$	A_k	$k \geq 2$	2
2		D,E,J,T,X,Z,W,Y		3
3		P,R,T,Q,S,U,V		50
> 3				105

Dabei läßt sich das k bei den A_k Singularitäten über die Milnorzahl bestimmen. Es bleiben also noch diejenigen vom corang 2 bzw. 3 zu betrachten.

3.2 Klassifikation für Corang 2

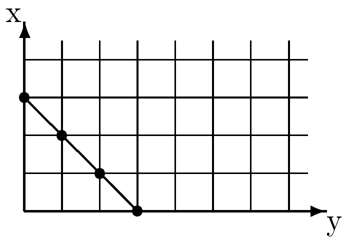
Satz 3.2

Für einen Keim $f \in \mathcal{O}_{(\mathbb{C}^2,0)}$ mit $f \in \mathcal{M}^3$ tritt einer der folgenden 4 Fälle ein:

Normalform	*Typ*	*d*	*m*	*Skizze*	*Schritt*
$j_3 f \approx\ x^3$	E_k, J	*1*	*2*		6_1
$j_3 f \approx\ x^2 y$	D_k	*1*	*1*		5
$j_3 f \approx\ x^2 y + y^3$	D_4	*2*			4
$j_3 f =\ 0$	X, Z, W, Y				13

wobei $d = dim(\mathbb{C}\{x,y\}/J_f) =$ `dim(std(`Jf`))` *und* m *gleich der Multiplizität des lokalen Ringes* $\mathbb{C}\{x,y\}/J_f$ *ist, d.h.* $m =$ `mult(std(`Jf`))`.

Das Newton Diagram zeigt, welche Monome in $j_3 f$ auftreten können. Es bleibt zu studieren, wie die Kombinationen der vier Monome aussehen können.



Aussehen von $j_3 f$

Beweis

Sei $f \in \mathcal{O}_{(\mathbb{C}^2,0)}$ mit $f \in \mathcal{M}^3$.
Betrachte nun den Fall $j_3 f \neq 0$ und schreibe $j_3 f$ wie folgt:
$j_3 f = a_0 x^3 + a_1 x^2 y + a_2 x y^2 + a_3 y^3 = (b_1 x + c_1 y)(b_2 x + c_2 y)(b_3 x + c_3 y)$.
Diese Zerlegung ist durchführbar, da $f \in \mathcal{O}_{(\mathbb{C}^2,0)}$ und $\mathbb{C}$ algebraisch abgeschlossen ist.
Wir können nun 3 Fälle vorfinden:

- 3 gleiche Faktoren $\Rightarrow$ Zeile 1 der Tabelle
- 2 gleiche Faktoren $\Rightarrow$ Zeile 2 der Tabelle
- 3 verschiedene Faktoren $\Rightarrow$ Zeile 3 der Tabelle

$\blacksquare$

Die **Funktion3** ist eine direkte Implementierung obiger Tabelle.

Für die Fälle 2 und 3 ist nun klar, um welche Singularität es sich dabei handelt; das k läßt sich über die Milnor-Zahl bestimmen. Es bleiben also noch Fall 1 und Fall 4 zu betrachten.

Betrachten wir zunächst $j_3 f \approx x^3$.
Hier muß als erstes ein Koordinatenwechsel φ gefunden werden, sodaß $j_3\varphi(f) = x^3$ gilt.

3.2.1 Koordinatentransformation für $(ax + by^k)^p$

Da eine solche Koordinatentransformation immer wieder an verschiedenen Stellen im Algorithmus vorkommt, betrachte dieses Problem etwas allgemeiner.
Sei dazu f ein Polynom in zwei Variablen mit folgender spezieller Gestalt:

$$f = \sum_{i+j=p} \binom{p}{j} a^i b^j x^i y^{jk} = (ax + by^k)^p$$

wobei f in expliziter Form sowie k und p gegeben sind.

Gesucht ist ein Koordinatenwechsel φ, sodaß $\varphi(f) = x^p$.
Die erste Syzygie von f und $\frac{\partial f}{\partial x}$ liefert ein Paar (r_1, r_2), so daß $(r_1, r_2) \cdot (f, \frac{\partial f}{\partial x})^T = 0$ ist.
D.h.:

$$\begin{aligned}
(r_1, r_2) \cdot (f, \frac{\partial f}{\partial x})^T = r_1 \cdot f + r_2 \cdot \frac{\partial f}{\partial x} = 0 \\
\Leftrightarrow r_1 \cdot (ax + by^k)^p = -pa \cdot r_2 \cdot (ax + by^k)^{p-1} \\
\Leftrightarrow r_1 \cdot (ax + by^k) = -pa \cdot r_2 \\
\Leftrightarrow r_1 \cdot (x + \frac{b}{a} y^k) = -p \cdot r_2
\end{aligned}$$

Für geeignete Wahl der Ordnung (c,ds) ist in Singular diese Syzygie minimal, d.h. $r_1 \neq 0$ vom Grad 0 und wir können φ wie folgt definieren:

$$\varphi : \begin{cases} y \mapsto y \\ x \mapsto x - \frac{b}{a} y^k, \end{cases}$$

wobei $\frac{b}{a}$ der Koeffizient von y^k in $\frac{-pr_2}{r_1}$ ist.
Dieser Algorithmus ist im Klassifizierer implementiert unter der Funktion Faktorisiere(poly f, poly g, int p, int k).
Hierbei ist f ein Polynom, so daß der gewichtete Jet $j_{x^p, y^{kp}}$ von f von der

Form $g = (ax + by^k)^p$ ist.

Bei der Beschreibung sind wir davon ausgegangen, daß $g \approx x^p$ gilt. Davon können wir aber im Allgemeinen nicht ausgehen. Es könnte ja durchaus auch $g \approx y^p$ gelten. Dieser Fall kann aber nur in den Schritten 6_1, 17, 25 vorkommen, weil hier $k = 1$ gilt und bis dahin die Rollen von x und y noch nicht fixiert worden sind.
Im nächsten Durchlauf des Algorithmus gilt z.B. bereits $j_{x^p, y^{k \cdot p}} f \approx x^p$ mit $k \geq 2$.
Wird eine solche Transformation zum ersten Mal bestimmt, so kann folgender Fall eintreten:

$$\begin{cases} \frac{\partial f}{\partial x} \neq 0 & \textit{oder} \\ \frac{\partial f}{\partial y} \neq 0 \end{cases}$$

Ansonsten gilt in jedem Fall $\frac{\partial f}{\partial x} \neq 0$

D.h. in den Fällen 6_1, 17, 25 muß eventuell eine Variablenvertauschung vorgenommen werden. In den anderen Fällen, in denen dieser Algorithmus angewandt wird, ist dies nicht nötig.

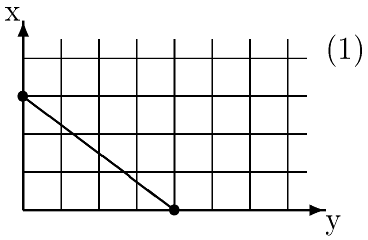
Im weiteren können wir nun davon ausgehen, daß $j_3 f = x^3$ ist. Zur weiteren Klassifikation müssen wir den "Gewichteten Jet" von f betrachten.

3.2.2 Schritt 6_k

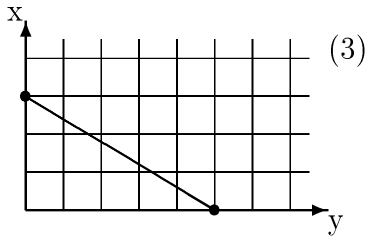
Im Schritt 6_k im Algorithmus von Arnold werden folgende vier Möglichkeiten unterschieden, wobei $k \in \mathbb{N}, k \geq 1$ ist.

Normalform	Typ	Schritt
$j_{x^3, y^{3k+1}} f \approx \; x^3 + y^{3k+1}$	E_{6k}	7_k
$j_{x^3, xy^{2k+1}} f \approx \; x^3 + xy^{2k+1}$	E_{6k+1}	8_k
$j_{x^3, y^{3k+2}} f \approx \; x^3 + y^{3k+2}$	E_{6k+2}	9_k
$j_{x^3, y^{3k+2}} f = \; x^3$		10_{k+1}

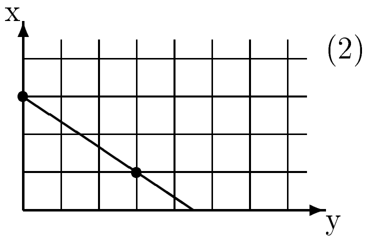
Zum Beweis betrachte dazu das Newton-Diagram von f. Wir erhalten den Gewichteten-Jet von f dadurch, daß wir eine Gerade in dieses Diagramm legen. Das Monom x^3 ist hier der Punkt, um den sich dann die Gerade nach und nach dreht; wir halten immer dann, wenn wir das nächste Monom treffen.



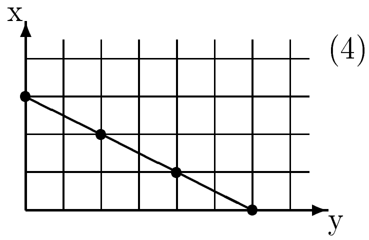
$j_{x^3,y^{3k+1}}f \approx \; x^3 + y^{3k+1}$ für $k = 1$



$j_{x^3,y^{3k+2}}f \approx \; x^3 + y^{3k+2}$ für $k = 1$



$j_{x^3,xy^{2k+1}}f \approx \; x^3 + xy^{2k+1}$ für $k = 1$



$j_{x^3,y^{3k}}f \approx \; x^3 + y^{3k}$ für $k = 2$

Die ersten 3 Bilder zeigen deutlich, daß es genügt die Existenz der einzelnen Monome im gewichtetem Jet zu prüfen. Damit ist der Typ der betreffenden Singularität bestimmt. Den Wert von k erhält man direkt aus dem Algorithmus. Eine Koordinatentransformation ist zur Bestimmung des Types der Singularität nicht mehr nötig. Allerdings hat man dadurch nicht die genaue Koordinatentransformation, die zur Normalform von f führt.

Im vierten Fall können bis zu vier Monome auftreten. Hier kann der Jet $j_{x^3,y^{3k}}f$ in 1, 2 oder 3 verschiedene Faktoren zerfallen, d.h. wir erhalten folgende Tabelle: (Schritt 10_k)

Normalform	Typ	d	m	Schritt
$j_{x^3,y^{3k}}f \approx \; x^3 + ax^2y^k + y^{3k}$ $(4a^3 + 27 \neq 0)$	$J_{k,0}$	0		11_k
$j_{x^3,y^{3k}}f \approx \; x^3 + x^2y^k$	$J_{k,p}$	1	1	12_k
$j_{x^3,y^{3k}}f \approx \; x^3$		1	2	6_k

wobei $d = dim(\mathbb{C}\{x, y\}/J_f) = \texttt{dim}(\texttt{std}(Jf))$ und m gleich der Multiplizität des lokalen Ringes $\mathbb{C}\{x, y\}/J_f$ ist, d.h. $m = \texttt{mult}(\texttt{std}(Jf))$.

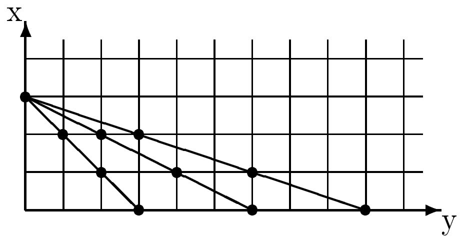
Dies läßt sich am einfachsten mit Hilfe von Dimension und Multiplizität des Jacobi-Ideals feststellen. Sind es 2 oder 3 verschiedene Faktoren, so haben wir den Typ gefunden, sonst muß wieder eine Koordinatentransformation φ gefunden werden, sodaß $j_{x^3,y^{3k}}f = \; x^3$gilt und es geht mit dem Schritt 6_k von

vorne los.
Diese Koordinatentransformation läßt sich, wie in 3.2.1 bereits beschrieben, allgemein bestimmen.

Die Klassifikation für $j_3 f \neq 0$ ist als Prozedur `Funktion6()` implementiert.

3.2.2.1 Zusammenfassung der Schritte 6_k bis 12

Folgendes Diagramm kann als Zusammenfassung der Schritte 6_k bis 12 betrachtet werden. Die Geraden zeigen an, an welchen Stellen mit Dimension, Multiplizität gearbeitet werden muß und ggf. eine Koordinatentransformation durchgeführt werden muß (mehr als 2 Monome auf der Geraden). Zwischen den Geraden bleiben hier jeweils nur drei Monome übrig, deren Existenz im Jet nur abgefragt werden muß, um den Typ bereits zu bestimmen.
Auf diesem Prinzip basiert der gesamte Klassifizierer, jedoch mit der Tendenz komplexer zu werden.



mögliche Monome in $j_{x^3,y^{3k}} f$ mit $k = 1, 2, 3, \ldots$

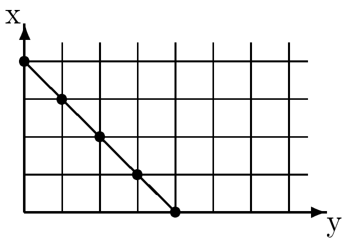
3.2.3 Schritt 13

Der **Schritt** 13 behandelt den Fall $j_3 f = 0$

Satz 3.3
Für einen Keim $f \in \mathcal{O}_{(\mathbb{C}^2,0)}$ mit $f \in \mathcal{M}^4$ gilt:

Normalform	*Typ*	d_1	*m*	d_2	*Skizze*	*Schritt*
$j_4 f \approx\ x^4 + ax^2y^2 + y^4, a^2 \neq 4$	X_9	*0*				14
$j_4 f \approx\ x^4 + x^2y^2$	$X_{1,p}$	*1*	*1*			15
$j_4 f \approx\ x^2y^2$	$Y^1_{p,q}$	*1*	*2*	*0*		16
$j_4 f \approx\ x^3y$	Z	*1*	*2*	*1*		17
$j_4 f \approx\ x^4$	W	*1*	*3*			25
$j_4 f =\ 0$						47

wobei $d_1 = dim(\mathbb{C}\{x,y\}/J_f) =$ `dim`(*`std`*(Jf)*)*, m gleich der Multiplizität des lokalen Ringes $\mathbb{C}\{x,y\}/J_f$ d.h $m =$ `mult`(`std`(Jf)) und $d_2 =$ `dim`(`std`(Jf, Hf)). H_f ist das Ideal, das von allen Einträgen der Hesse-Matrix erzeugt wird.*



Aussehen von $j_{x^4} f$

Beweis

Sei $f \in \mathcal{O}_{(\mathbb{C}^2,0)}$ mit $f \in \mathcal{M}^3$.
Betrachte nun den Fall $j_4 f \neq 0$ und schreibe f wie folgt:
$j_3 f = a_0x^4 + a_1x^3y + a_2x^2y^2 + a_2xy^3 + a_4y^4 = (b_1x + c_1y)(b_2x + c_2y)(b_3x + c_3y)(b_4x + c_4y)$
Diese Zerlegung ist durchführbar, da $f \in \mathcal{O}_{(\mathbb{C}^2,0)}$ und $\mathbb{C}$ algebraisch abgeschlossen.
Wir können nun 5 Fälle vorfinden:

- 4 verschiedene Faktoren $\Rightarrow$ Zeile 1 der Tabelle
- 2 gleiche Faktoren und 2 verschiedene $\Rightarrow$ Zeile 2 der Tabelle
- 2 mal 2 gleiche Faktoren $\Rightarrow$ Zeile 3 der Tabelle
- 3 gleiche Faktoren $\Rightarrow$ Zeile 4 der Tabelle
- 1 Faktor $\Rightarrow$ Zeile 5 der Tabelle

■

Nach dem Arnold-Algorithmus sind für die ersten drei Fälle die Klassen bereits eindeutig. Bei der $X_{1,p}$ erhält man das p über die Milnorzahl, da $\mu(X_{1,p}) = 9 + p$ ist. Bei $Y^1_{p,q}$ sind jedoch zwei Indizes zu bestimmen. Da hier die Milnorzahl $\mu(Y^1_{p,q}) = 9 + p + q$, erhält man diese derzeit nur über den Hilbert-Klassifizierer(s. Anhang A). Die drei anderen Fälle müssen weiter klassifiziert werden.

Der Fall $j_3 f = 0$ mit den ersten 3 Zeilen der Tabelle in 3.3 ist als Prozedur Funktion13() implementiert.

3.2.3.1 Schritt 17

Es wird der Fall $j_4 f \approx x^3 y$ behandelt.

D.h. $j_4 f = (ax+by)^3 \cdot (cx+dy)$, wobei a, b, c und d noch zu bestimmen sind.
Nach Voraussetzung gilt: $det \begin{pmatrix} a & b \\ c & d \end{pmatrix} = ad - bc \neq 0$
Notation: $g := j_4 f$.

Ausmultiplizieren liefert:

$$g = a^3cx^4 + a^2(3bc+ad)x^3y + ab(3bc+3ad)x^2y^2 + b^2(bc+3ad)xy^3 + b^3dy^4$$

Und damit erhalten wir unter berücksichtigung von $ad - bc \neq 0$ folgende Tabelle mit 7 Fällen:

Koeffizient von				Konsequenz für				Fall	Transformation
x^4	y^4	x^3y	xy^3	a	b	c	d		
0	0	$\neq 0$	0	$\neq 0$	0	0	$\neq 0$	$\varepsilon x^3 y$	
0	0	0	$\neq 0$	0	$\neq 0$	$\neq 0$	0	εxy^3	
0	$\neq 0$	0	$\neq 0$	0	$\neq 0$	$\neq 0$	$\neq 0$	$\varepsilon y^3(cx+dy)$	$syz(f, \frac{\partial f}{\partial x})$
$\neq 0$	0	$\neq 0$	0	$\neq 0$	0	$\neq 0$	$\neq 0$	$\varepsilon x^3(cx+dy)$	$syz(f, \frac{\partial f}{\partial y})$
0	$\neq 0$	$\neq 0$		$\neq 0$	$\neq 0$	0	$\neq 0$	$\varepsilon(ax+by)^3 y$	$syz(f, \frac{\partial f}{\partial x})$
$\neq 0$	0		$\neq 0$	$\neq 0$	$\neq 0$	$\neq 0$	0	$\varepsilon(ax+by)^3 x$	$syz(f, \frac{\partial f}{\partial y})$
$\neq 0$	$\neq 0$			$\neq 0$	$\neq 0$	$\neq 0$	$\neq 0$	$(ax+by)^3(cx+dy)$	

- Fall 1: $j_4 f = a^3 d x^3 y$, ist schon in der gesuchten Form.

- Fall 2: $j_4 f = b^3 cxy^3$ wähle $\varphi : \begin{cases} x \mapsto y \\ y \mapsto x \end{cases}$

- Fall 3: $j_4 f = b^3 y^3 \cdot (cx+dy)$

  liefert die Berechnung von $syz(g, \frac{\partial g}{\partial x}) = (r_1, r_2)$:

  $$r_1 = c, r_2 = -(cx+dy)$$

Damit erhalte $\varphi : \begin{cases} x \mapsto y - \frac{d}{c}x \\ y \mapsto x \end{cases}$

- Fall 4: $j_4 f = a^3 x^3 \cdot (cx + dy)$

  Analog zu Fall 3, nur mit vertauschten Variablen.
  Damit erhalte $\varphi : \begin{cases} x \mapsto x \\ y \mapsto y - \frac{c}{d}x \end{cases}$

- Fall 5: $j_4 f = (ax + by)^3 \cdot dy$

  liefert die Berechnung von $syz(g, \frac{\partial g}{\partial x}) = (r_1, r_2)$:

  $$r_1 = 3a, r_2 = -(ax + by)$$

  Damit erhalte $\varphi : \begin{cases} x \mapsto x - \frac{b}{a}y \\ y \mapsto y \end{cases}$

- Fall 6: $j_4 f = (ax + by)^3 \cdot cx$

  Analog zu Fall 5, nur mit vertauschten Variablen.
  Damit erhalte $\varphi : \begin{cases} x \mapsto y \\ y \mapsto x - \frac{a}{b}y \end{cases}$

- Fall 7: $j_4 f = (ax + by)^3 \cdot (cx + dy)$.

  Reduktion auf einen der vorherigen Fälle.
  Die Berechnung von $syz(g, \frac{\partial g}{\partial x}) = (r_1, r_2)$ und $syz(g, \frac{\partial g}{\partial y}) = (r_1', r_2')$ liefert:

  $$r_1 = \underbrace{-(4ac)}_{=:A} x - \underbrace{(3ad + bc)}_{=:B} y$$

  $$r_2 = \underbrace{ac}_{=:\alpha} x^2 + \underbrace{(ad + bc)}_{=:\beta} xy + \underbrace{bd}_{=:\gamma} y^2 = r_2'$$

  $$r_1' = \underbrace{-(3bc + ad)}_{=:C} x - \underbrace{4bd}_{=:D} y$$

  $\Rightarrow ad = \frac{(B-\beta)}{2}$ und $bc = \frac{(C-\beta)}{2} \Rightarrow \frac{b}{a} = \frac{2\gamma}{B-\beta}$, analog folgt $\frac{c}{d} = \frac{C-\beta}{2\gamma}$

  Dann liefert die Koordinatentransformation $\varphi : \begin{cases} x \mapsto x - \frac{b}{a}y \\ y \mapsto y - \frac{c}{d}x \end{cases}$

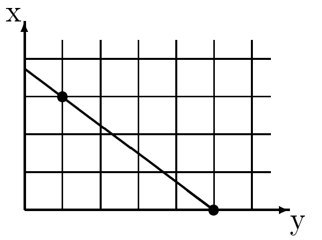
$$\varphi(j_4 f) = (a(x - \tfrac{b}{a}y) + b(y - \tfrac{c}{d}x))^3 \cdot (c(x - \tfrac{b}{a}y) + d(y - \tfrac{c}{d}x))$$
$$= (a - \tfrac{bc}{d})^3 (d - \tfrac{bc}{a}) x^3 y$$

Die Implementierung des obiges Verfahrens ist die Prozedur Isomorphie_s17().

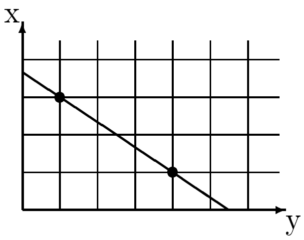
Nachdem nun $j_4 f = \varepsilon x^3 y$ gilt, können wir zum Schritt 18_1 übergehen.

3.2.3.2 Schritt 18_p

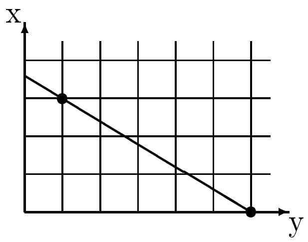
Nun können folgende vier Fälle mit Hilfe des Newtondiagramms unterschieden werden:



$j_{x^3y,y^{3p+2}} f \approx \; x^3 y + y^{3p+2}$ für $p = 1$



$j_{x^3y,xy^{2p+2}} f \approx \; x^3 y + x y^{2p+2}$ für $p = 1$



$j_{x^3y,y^{3p+3}} f \approx \; x^3 y + y^{3p+3}$ für $p = 1$

Normalform	Typ	Schritt
$j_{x^3y,y^{3p+2}} f \approx \; x^3 y + y^{3p+2}$	Z_{6p+5}	19_p
$j_{x^3y,xy^{2p+2}} f \approx \; x^3 y + x y^{2p+3}$	Z_{6p+6}	20_p
$j_{x^3y,y^{3p+3}} f \approx \; x^3 y + y^{3p+3}$	Z_{6p+7}	21_p
$j_{x^3y,y^{3p+3}} f = \; x^3 y$		22_{p+1}

3.2.3.3 Schritt 22_p

Für die folgenden drei Fälle findet die Unterscheidung wieder mit Hilfe von Invarianten statt.

Normalform	Typ	d	m	Schritt
$j_{x^3y,y^{3p+1}}f \approx\ y(x^3+bx^2y^p+y^{3p})$, $4b^3+27\neq 0$	$Z_{p-1,0}$	0		23_p
$j_{x^3y,y^{3p+1}}f \approx\ y(x^3+x^2y^p)$	$Z_{p-1,r}$	1	1	24_p
$j_{x^3y,y^{3p+1}}f \approx\ x^3y$		1	2	18_p

wobei $d = dim(\mathbb{C}\{x,y\}/J_f) = \texttt{dim}(\texttt{std}(Jf))$ und m gleich der Multiplizität des lokalen Ringes $\mathbb{C}\{x,y\}/J_f$ ist, d.h. $m = \texttt{mult}(\texttt{std}(Jf))$.

Für den Übergang zu Schritt 18_p muß wieder eine Koordinatentransformation ermittelt werden, so daß $j_{x^3y,y^{3p+1}}f =\ x^3y$ gilt.

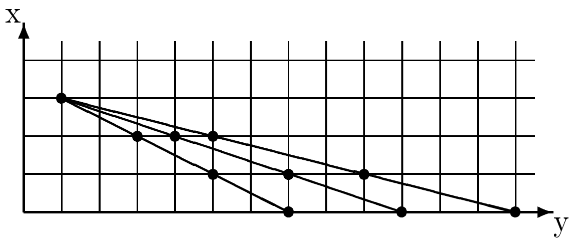
Wir haben $j_{x^3y,y^{3p+1}}f =\ (ax+by^p)^3y$.
Setze also $g := \frac{j_{x^3y,y^{3p+1}}f}{y}$ und berechne die Syzygien von $(g, \frac{\partial g}{\partial x})$. So wie bereits in 3.2.1 beschrieben erhalten wir dann die Koorinatentransformation

$$\varphi : \begin{cases} y \mapsto y \\ x \mapsto x - \frac{b}{a}y^p \end{cases}$$

die das Gewünschte liefert.

3.2.3.4 Zusammenfassung der Schritte 17 bis 24

(vergleiche. 3.2.2.1)



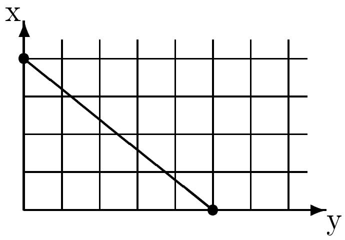
Mögliche Monome für $j_{x^3,y^{3k}}f$ mit $k = 1, 2, 3\ldots$

3.2.4 Schritt 25

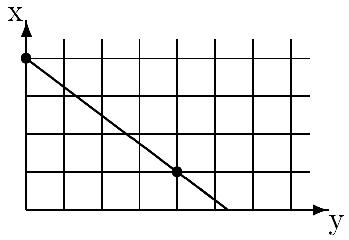
3.2.4.1 Schritt 25 / 26_1

Als erstes muß ein Koordinatenwechsel gefunden werden so, daß $j_4f = x^4$. Dieser läßt sich mit dem in 3.2.1 bereits beschriebenen Verfahren bestimmen. Anschließend (Schritt 26_1) können folgende drei Fälle mit Hilfe des Newtondiagramms unterschieden werden:

Normalform	Typ	Schritt
$j_{x^4,y^{4k+1}}f \approx\ x^4 + y^{4k+1}$	W_{12k}	27_k
$j_{x^4,xy^{3k+1}}f \approx\ x^4 + xy^{3k+1}$	W_{12k+1}	28_k
$j_{x^4,xy^{3k+1}}f =\ x^4$		29_k



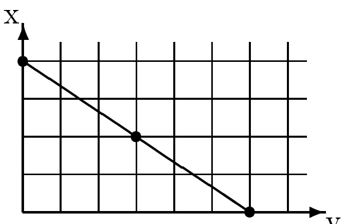
$j_{x^4,y^{4k+1}}f \approx\ x^4 + y^{4k+1}$ für $k = 1$



$j_{x^4,xy^{3k+1}}f \approx\ x^4 + xy^{3k+1}$ für $k = 1$

3.2.4.2 Schritt 29_k

Der dritte Fall, bei dem bis zu 3 Monome auftreten können, läßt mit Hilfe von `dim`() und `mult`() wieder vier Möglichkeiten zu.



$j_{x^4,y^{4k+2}}f \approx\ x^4$ für $k = 1$

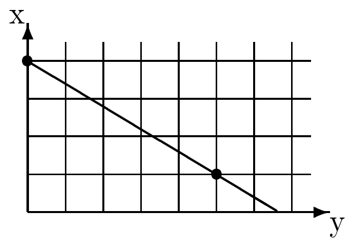
Normalform	Typ	d	m	Schritt
$j_{x^4,y^{4k+2}}f \approx\ x^4 + bx^2y^{2k+1} + y^{4k+2}\ ,b^2 \neq 4$	$W_{k,0}$	0		30_k
$j_{x^4,y^{4k+2}}f \approx\ (x^2 + y^{2k+1})^2$	$W^{\#}_{k,i}$	1	2	32_k
$j_{x^4,y^{4k+2}}f \approx\ x^4 + x^2y^{2k+1}$	$W_{k,i}$	1	1	31_k
$j_{x^4,y^{4k+2}}f =\ x^4$		1	3	33_k

wobei $d = dim(\mathbb{C}\{x,y\}/J_f) = \mathtt{dim}(\mathtt{std}(Jf))$ und m gleich der Multiplizität des lokalen Ringes $\mathbb{C}\{x,y\}/J_f$ ist, d.h. $m = \mathtt{mult}(\mathtt{std}(Jf))$.

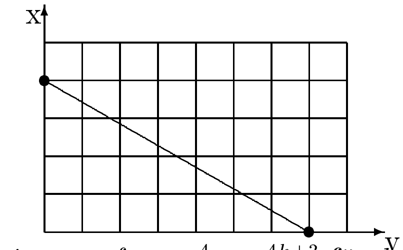
3.2.4.3 Schritt 33_k

Da für den Fall 4 keine Koordinatentransformation mehr nötig ist, können mit Hilfe des Newton-Diagramms folgende 3 Möglichkeiten unterschieden werden.

Normalform	Typ	Schritt
$j_{x^4,xy^{3k+2}}f \approx\ x^4 + xy^{3k+2}$	W_{12k+5}	34_k
$j_{x^4,y^{4k+3}}f \approx\ x^4 + y^{4k+3}$	W_{12k+6}	35_k
$j_{x^4,y^{4k+3}}f =\ x^4$		36_{k+1}



$j_{x^4,xy^{3k+2}}f \approx\ x^4 + xy^{3k+2}$ für $k = 1$



$j_{x^4,y^{4k+3}}f \approx\ x^4 + y^{4k+3}$ für $k = 1$

3.2.4.4 Schritt 36_{k+1}

Wenn wir für den letzten Fall die Gerade durch x^4 und y^{4k+3} um x^4 etwas weiter drehen, müssen wir den Jet $j_{x^4,y^{4k}}f$ betrachten. Für diesen gibt es nun 5 Möglichkeiten. Sie lassen sich diesmal mit Hilfe von d, m unterscheiden. (Durch die Koordinatentransformation $x \mapsto x - \alpha y^k$ läßt sich ein Term cy^{4k} entfernen).

Normalform	Typ	d	m	ds	Schritt
$j_{x^4,y^{4k}}f \approx\ x^4 + bx^3y^k + ax^2y^{2k} + xy^{3k}\ ,\Delta \neq 0,\ ab \neq 9$	$X_{k,0}$	0			37_k
$j_{x^4,y^{4k}}f \approx\ x^2(x^2 + axy^k + y^{2k})\ ,\ a^2 \neq 4$	$X_{k,p}$	1	1		38_k
$j_{x^4,y^{4k}}f \approx\ x^2(x + y^k)^2$	$Y^k_{r,s}$	1	2	1	39_k
$j_{x^4,y^{4k}}f \approx\ x^3(x + y^k)$	$f \sim f_1f_2$	1	2	0	40_k
$j_{x^4,y^{4k}}f \approx\ x^4$		1	3		26_k

wobei $d = dim(\mathbb{C}\{x,y\}/J_f) =$ `dim(std(`Jf`))`, m gleich der Multiplizität des lokalen Ringes $\mathbb{C}\{x,y\}/J_f$ ist, d.h. $m =$ `mult(std(`Jf`))` und $ds =$ `dim(`$\mathrm{J}_{j_{x^4,y^{4k}}f/x^2}$`)`,$\Delta = 4(a^3 + b^3) + 27 - a^2b^2 - 18ab$.

Der Fall 5 bildet den Schluß unserer Schleife. Nach Finden und Ausführen der passenden Koordinatentransformation (vergleiche. 3.2.1) steigen wir wieder bei Schritt 26_k ein.

Für den Fall 4 folgt:

3.2.4.5 Schritt 40_k

Schritt 40_k besagt, daß sich f schreiben läßt als Produkt zweier Polynome f_1 und f_2 mit $j_{x,y^k}f_1 \approx x + y^k$ und $j_{x^3,y^{3k}}f_2 = x^3$. Es gilt $k \geq 2$.

Zunächst muß jedoch ersteinmal eine Koordinatentransformation gefunden werden, so daß $j_{x^4,y^{4k}}f = x^3(x + y^k)$ gilt.
Nehme dazu an, daß $j_{x^4,y^{4k}}f = (ax + by)^4 + (ax + by)^3(cx + dy)^k$ mit $k \geq 2$ und $ad - bc \neq 0$ ist. Ausmultiplizieren liefert einen Term b^4y^4. Jedoch gilt $j_{x^4,y^{4(k-1)+3}}f = x^4$. Damit ist $b = 0$.
Der Jet von f kann also lediglich von folgender Form sein:

$$j_{x^4,y^{4k}}f = (ax)^4 + (ax)^3(cx + dy)^k$$

Ausrechnen liefert: $j_{x^4,y^{4k}}f = (ax)^4 + (ax)^3(\sum\limits_{i+j=k} \binom{k}{j} c^i d^j x^i y^j)$.

Mit den Gewichten (w_1, w_2)= $(k\ , 1)$ für (x,y) erhalten wir folgende Grade für die Monome:

x^4	x^3y^k	x^3xy^{k-1}	$x^3x^2y^{k-2}$	$\dots$	$x^3x^{k-1}y$	x^3x^k
$4k$	$4k$	$5k-1$	$6k-2$	$\dots$	k^2+2k+1	k^2+3k

Da $k \geq 2$ folgt $j_{x^4,y^{4k}}f = \alpha x^4 + \beta x^3y^k$ mit $\alpha \neq 0$ und $\beta \neq 0$ (denn jedes Monom von $j_{x^4,y^{4k}}f$ hat dem gewichteten Grad $4k$).

Damit folgt $c = 0$, $\beta = a^3d^k \neq 0$.
Anschließend können wir auf die Suche nach f_1 und f_2 gehen, so daß gilt $f = f_1 \cdot f_2$, mit $j_{x,y^k}f_1 \approx x + y^k$ und $j_{x^3,y^{3k}}f_2 = x^3$.
In der Implementierung der Syzygien in Singular liefert uns $\mathtt{syz}(f - j_{x^4,y^{4k}}f, x^3, \alpha x + \beta y^k)$ mit α, $\beta \neq 0$ folgendes Gleichungssystem (was sich auch leicht theoretisch einsehen läßt):

$$\begin{pmatrix} (r_{11} = 1) & r_{21} & r_{31} \\ (r_{12} = 0) & r_{22} & r_{32} \end{pmatrix} \cdot \begin{pmatrix} f - j_{x^4,y^{4k}}f \\ x^3 \\ \alpha x + \beta y^k \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Aus der zweiten Zeile folgt: $r_{32} = -x^3$ und $r_{22} = \alpha x + \beta y^k$.
Die erste Zeile liefert:

$$f - x^3(\alpha x + \beta y^k) + r_{21} \cdot x^3 + r_{31} \cdot (\alpha x + \beta y^k) = 0$$

$$\Leftrightarrow f = x^3(\alpha x + \beta y^k) - r_{21} \cdot x^3 - r_{31} \cdot (\alpha x + \beta y^k)$$

Mit der Zeile 2 erhalten wir:

$$f = -r_{32} \cdot r_{22} + r_{21} \cdot r_{32} - r_{31} \cdot r_{22}$$
$$f = (r_{21} - r_{22}) \cdot (r_{31} + r_{32}) - r_{31} \cdot r_{21}$$

Bestimmen wir zunächst das Aussehen von r_{21} und r_{31}:

Die erste Zeile unserer Syzygienmatrix liefert:

$$f - j_{x^4,y^{4k}} f + r_{21} \cdot x^3 + r_{31} \cdot (\alpha x + \beta y^k) = 0$$

$$\Rightarrow j_{x^4,y^{4k}}(r_{21} \cdot x^3 + r_{31} \cdot (\alpha x + \beta y^k)) = 0$$

Da $\beta \neq 0$ ist, kann r_{21} keinen konstanten Term haben und damit muß $j_{x^2,y^{3k-1}} r_{31} = 0$ sein (Nach Konstruktion).

Es bleiben zwei Möglichkeiten zu betrachten:

- $j_{x^3,y^{3k}} r_{31} \neq 0$: Dann gilt: $j_{x^3,y^{3k}} r_{31} = -x^3$ und $j_{x,y^k} r_{21} = \alpha x + \beta y^k$. damit ergibt sich: $f = r_{31} \cdot r_{21}$ Setze also $f_1 = r_{21}$ und $f_2 = r_{31}$.

- $j_{x^3,y^{3k}} r_{31} = 0$, so folgt auch $j_{x,y^k} r_{21} = 0$.
  Gibt es nun eine Koordinatentransformation φ, so daß

  $$\varphi(f) = (r_{21} - r_{22}) \cdot (r_{31} + r_{32}),$$

  so ist auch hier unser Ziel erreicht.
  Zunächst gehen wir davon aus, daß es eine solche Koordinatentransformation gibt (sie wird aber nicht ausgeführt).
  Setze $f_1 = r_{21} - r_{22}$ und $f_2 = r_{31} + r_{32}$.

In beiden obigen Fällen gilt: $j_{x,y^k} f_1 \approx x + y^k$ und $j_{x^3,y^{3k}} f_2 = \varepsilon x^3$.
Unter der Voraussetzung, daß φ existiert, genügt es, um f zu klassifizieren, den Typ von f_2 zu bestimmen. Wende dazu den schon behandelten Klassifizierer auf f_2 an und erhalte folgende Tabelle:

f_2	Schritt	f
$f_2 \in E_{6(k+i)}$, $k \geq 2$, $i \geq 0$	$42_{k,i}$	$f \in Z^k_{12k+6i-1}$
$f_2 \in E_{6(k+i)+1}$, $k \geq 2$, $i \geq 0$	$43_{k,i}$	$f \in Z^k_{12k+6i}$
$f_2 \in E_{6(k+i)+2}$, $k \geq 2$, $i \geq 0$	$44_{k,i}$	$f \in Z^k_{12k+6i+1}$
$f_2 \in J_{k+i+1,0}$, $k \geq 2$, $i \geq 1$	$45_{k,i+1}$	$f \in Z^k_{i,0}$
$f_2 \in J_{k+i+1,p}$, $k \geq 2$, $i \geq 1$, $p > 0$	$46_{k,i+1,p}$	$f \in Z^k_{i,p}$

Es bleibt zu zeigen, daß die Koordinatentransformation φ existiert. Notiere dazu die kleinsten Monome die in r_{21}, r_{31} und $r_{21} \cdot r_{31}$ auftreten können.

r_{21}	x^2	xy^ν	y^{k+1}				
r_{31}	x^4	$x^3y^{\nu'}$	$x^2y^{k+\nu'}$	$xy^{2k+\nu'}$	y^{3k+1}		
$r_{21} \cdot r_{31}$	x^6	$x^5y^{\nu'}$	$x^4y^{k+\nu'}$	$x^3y^{2k+\nu'}$	x^2y^{3k+1}		
		x^5y^ν	$x^4y^{\nu+\nu'}$	$x^3y^{k+\nu+\nu'}$	$x^2y^{2k+\nu+\nu'}$	$xy^{3k+\nu+1}$	
			x^4y^{k+1}	$x^3y^{k+\nu'+1}$	$x^2y^{2k+\nu'+1}$	$xy^{3k+\nu'+1}$	y^{4k+2}
$r_{21} \cdot r_{31}$	x^6	x^5y^ν	$x^4y^{\nu+\nu'}$	$x^3y^{k+\nu'+1}$	$x^2y^{2k+\nu'+1}$	$xy^{3k+\nu'+1}$	y^{4k+2}

Analog für $f_1 = r_{21} - r_{22}$, $f_2 = r_{31} + r_{32}$ und $f_1 \cdot f_2$.

$r_{21} - r_{22}$	x^2	αx	xy^ν	βy^k	y^{k+1}			
$r_{31} + r_{32}$	x^4	x^3	$x^3y^{\nu'}$	$x^2y^{k+\nu'}$	$xy^{2k+\nu'}$	y^{3k+1}		
$f_1 \cdot f_2$	x^6	x^5	$x^5y^{\nu'}$	$x^4y^{k+\nu'}$	$x^3y^{2k+\nu'}$	x^2y^{3k+1}		
		x^5	x^4	$x^4y^{\nu'}$	$x^3y^{k+\nu'}$	$x^2y^{2k+\nu'}$	xy^{3k+1}	
		x^5y^ν	x^4y^ν	$x^4y^{\nu+\nu'}$	$x^3y^{k+\nu+\nu'}$	$x^2y^{2k+\nu+\nu'}$	$xy^{3k+\nu+1}$	
			x^4y^k	x^3y^k	$x^3y^{k+\nu'}$	$x^2y^{2k+\nu'}$	$xy^{3k+\nu'}$	y^{4k+1}
			x^4y^{k+1}	x^3y^{k+1}	$x^3y^{k+\nu'+1}$	$x^2y^{2k+\nu'+1}$	$xy^{3k+\nu'+1}$	y^{4k+2}
$f_1 \cdot f_2$	x^4	x^3y^k	x^5	x^4y^ν	$x^3y^{k+\nu'}$	$x^2y^{2k+\nu'}$	$xy^{3k+\nu'}$	y^{4k+1}

Jeweils mit $1 \leq \nu \leq k$ und $1 \leq \nu' \leq k$.

Mache nun den Ansatz: $f = f_1 \cdot f_2 - r_{21} \cdot r_{31} \sim g_1 \cdot g_2$
d.h.: $\varphi(g_1 \cdot g_2) = \underbrace{\varphi(g_1)}_{x+y^k+\dots} \cdot \underbrace{\varphi(g_2)}_{x^3+\dots} = f_1 \cdot f_2 - r_{21} \cdot r_{31}$

$r_{21} \cdot r_{31}$ läßt sich wie folgt schreibe:
$x^3 \cdot \underbrace{(*x^3 + *x^2y^\nu + *xy^{\nu+\nu'} + *y^{k+\nu+1})}_{wird\ von\ \varphi(g_1)\ erschlagen} + *\, x^2y^{2k+\nu'+1} + *xy^{3k+\nu'+1} + *y^{4k+2}$

Wir müssen nun noch die letzten drei Terme betrachten. Schreibe dazu noch einmal ausführlich, $*$ steht hierbei für einen beliebigen Koeffizienten aus $\mathbb{C}$:

$$f_1 = r_{21} - r_{22} = -\alpha x - \beta y^k + (*x^2 + *xy^\nu + *y^{k+1})$$
$$f_2 = r_{31} + r_{32} = -x^3 + (*x^4 + *x^3y^{\nu'} + *x^2y^{k+\nu'} + *xy^{2k+\nu'} + *y^{3k+1})$$

$f = -r_{22} \cdot r_{32} + r_{21} \cdot r_{32} - r_{22} \cdot r_{31} =$
$x^3(\alpha x + \beta y^k) - x^3(*x^2 + *xy^\nu + *y^{k+1}) - (\alpha x + \beta y^k)(*x^4 + *x^3 y^{\nu'} + *x^2 y^{k+\nu'} + *xy^{2k+\nu'} + *y^{3k+1})$
$f_1 \cdot f_2 = f - r_{21} \cdot r_{31}$

Der Term $*x^2 y^{2k+\nu'+1}$ läßt sich mittels $x \mapsto x - *y^{2k+\nu'+1}$ eliminieren. Es bleiben also noch die letzten zwei Terme übrig.

- $*y^{4k+2}$: Gilt $* \neq 0$ so existiert in f, wegen $\beta \neq 0$, bereits $\beta y^k \cdot ay^{3k+1}$ mit $a \neq 0$. Spätestens dieser Term legt den Typ von f_2 fest, so daß $*y^{4k+2}$ keinen Einfluß mehr darauf haben kann.

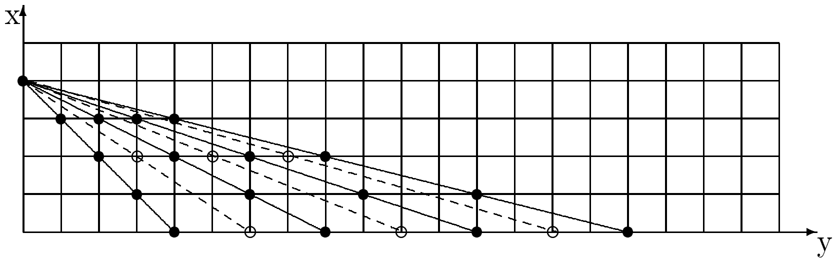
- $*xy^{3k+\nu+1}$ entsteht aus:

  - $*xy^\nu \cdot ay^{3k+1}$: Gilt $* \neq 0$ so existiert in f, wegen $\alpha \neq 0$, bereits $\alpha x \cdot ay^{3k+1}$ mit $a \neq 0$. Spätestens dieser Term legt den Typ von f_2 fest, so daß $*xy^{3k+\nu+1}$ keinen Einfluß mehr darauf haben kann.
  - $*y^{k+1} \cdot bxy^{2k+\nu'}$: Gilt $* \neq 0$ so existiert in f, wegen $\beta \neq 0$, bereits $\beta y^k \cdot bxy^{2k+\nu'}$ mit $b \neq 0$. Spätestens dieser Term legt den Typ von f_2 fest, so daß $*xy^{3k+\nu'+1}$ keinen Einfluß mehr darauf haben kann.

Damit ist gezeigt, daß also $f \sim (r_{21} - r_{22}) \cdot (r_{31} + r_{32})$.
Und dies reicht für unsere Zwecke aus.

3.2.4.6 Zusammenfassung der Schritte 25 bis 46

Möglichkeiten für $j_{x^4,y^{4k}}f$ und $j_{x^4,y^{4k+2}}f$



Mit jeder durchgezogenen Linie beginnt ein Schleifendurchlauf. Bei jeder gestrichelte Linie (Schritt 29_k) muß eine Koordinatentransformation bestimmt und durchgeführt werden.(vergleiche die Bemerkung in 3.2.2.1)

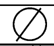
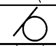
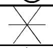
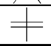
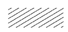
3.2.5 Schritt 47

Der **Schritt 47** behandelt den Fall $j_4 f = 0$ und ist im Programm nicht weiter verfolgt. Ist aber leicht: $j_5 f = g$ $\dim = 0 \Rightarrow f \in N_{16}$
$\dim = 1$ $j_5 f$ entartet, $\mu > 10$, $m > 2$, $c > 12$.

3.3 Klassifikation für Corang 3

Die Klassifikation in Corang 3 ist ab dem Schritt 50 durchgeführt. Der Aufbau der Klassifikation ist ähnlich wie in Corang 2, was die Fallunterscheidungen betrifft, die Kriterien sind auch hier Dimension, Multiplizität, Gewichteter Jet und Newton Diagramm; jedoch ist das Finden der Koordinatentransformationen mit grösseren Schwierigkeiten verbunden. Denn im Gegensatz zu Corang 2 haben wir nun die Wahl unter drei Variablen, in denen die erste Koordinatentransformation durchzuführen ist. Es mussten also andere Wege gefunden werden, um die Rollen der einzelnen Koordinaten x, y und z zu fixieren. Darin scheiterte bis jetzt auch eine vollständige Implementierung dieses Teils des Algorithmus.

Satz 3.4
Sei $f \in \mathbb{C}[x, y, z]$ homogen vom Grad 3. $f \neq 0$. Dann ist f rechtsäquivalent zu einem der 10 Fälle:

Normalform	*Typ*	d_1	m	d_2	*Skizze*	*Schritt*
$f \approx x^3 + y^3 + z^3 + axyz$	(P_8)	*0*	*8*	*0*		*51*
$f \approx x^3 + y^3 + xyz$	(P)	*1*	*1*	*0*		*52*
$f \approx x^3 + xyz$	(R)	*1*	*2*	*0*		*54*
$f \approx xyz$	(T)	*1*	*3*	*0*		*56*
$f \approx x^3 + yz^2$	(Q)	*1*	*2*	*1*		*58*
$f \approx x^2 z + yz^2$	(S)	*1*	*3*	*1*		*66*
$f \approx x^3 + xz^2$	(U)	*1*	*4*	*1*		*82*
$f \approx x^2 y$	(V)	*2*	*1*	*1*		*97*
$f \approx x^3$	(V)	*2*	*2*	*2*		*103*
$f = 0$	(V)					*104*

wobei $d_1 = dim(\mathbb{C}\{x, y, z\}/J_f) =$ `dim(std(`Jf`))`, *m gleich der Multiplizität des lokalen Ringes $\mathbb{C}\{x, y, z\}/J_f$ ist d.h* $m =$ `mult(std(`Jf`))` *und*
$d_2 =$ `dim(std(wedge(jacob(`Jf`),2),`Jf`))`, $a^3 + 27 \neq 0$.

Wir wissen nun zu welcher Normalform unser 3-Jet isomorph ist, wir kennen jedoch noch nicht die zugehöhrige Koordinatentransformation für die Fälle, in denen anschliessend der Algorithmus noch weiter geht.
Diese gilt es jetzt also zu finden.

3.3.1 Schritte 51 bis 57

Im ersten Fall $f \approx P_8 = T_{3,3,3}$ hat man bereits alle Informationen. Bei $f \approx P_{p+5} = T_{3,3,3+p}$ erhält man das p mit Hilfe der Milnorzahl. Für die nächsten zwei Fälle $f \approx R_{p,q} = T_{3,p,q}$, $f \approx T_{p,q,r}$ genügt jedoch die Milnorzahl nicht. Die zwei bzw. drei Indizes können hier nur mit Hilfe des Milnorcodes bestimmt werden.

3.3.2 Schritt 58

Lege im ersten Teilschritt die Singularität in den Punkt $(0,0,1)$, d.h transformiere f, so daß $j_3 f = z f_2(x,y) + f_3(x,y)$.
Sei dazu $S := sat(J_f)$ das gesätigte Jacobi-Ideal. Berechne nun von $S_1 := \frac{\partial S}{\partial x_i}$ die 2x2 Minoren und sättige auch dieses Ideal. Dieses Ideal wird von zwei linearen Polynomen erzeugt. Schreibe die Koeffizienten in eine Matrix, berechne davon die Determinante, und erhalte damit die Koordinaten (b_1, b_2, b_3) des Singulären Ortes. Verschieben in den Punkt $(0,0,1)$ liefert dann das gewünschte Ergebnis.
Sei A die Matrix der Koordinatentransformation.

- $b_1 \neq 0$: $A = \begin{pmatrix} b_1 & 0 & 0 \\ b_2 & 1 & 0 \\ b_3 & 0 & 1 \end{pmatrix}$ $\varphi_1 : \begin{cases} x \mapsto -b_1 x \\ y \mapsto -b_2 x + y \\ z \mapsto -b_3 x + z \end{cases}$

- $b_2 \neq 0$: $A = \begin{pmatrix} 1 & b_1 & 0 \\ 0 & b_2 & 0 \\ 0 & b_3 & 1 \end{pmatrix}$ $\varphi_1 : \begin{cases} x \mapsto x - b_1 y \\ y \mapsto -b_2 y \\ z \mapsto -b_3 y + z \end{cases}$

- $b_3 \neq 0$: $A = \begin{pmatrix} 1 & 0 & b_1 \\ 0 & 1 & b_2 \\ 0 & 0 & b_3 \end{pmatrix}$ $\varphi_1 : \begin{cases} x \mapsto x - b_1 z \\ y \mapsto y - b_2 y \\ z \mapsto -b_3 z \end{cases}$

Wir beschränken uns in der folgenden Erläuterung auf den Fall $b_1 \neq 0$; implementiert sind alle drei Fälle.

Nach der Anwendung von φ_1 hat unser 3-Jet die Form: $x f_2(y,z) + f_3(y,z)$.

Bestimme im zweiten Teilschritt zunächst, sofern nötig, f_2. Prüfe dazu, ob beide partiellen Ableitungen verschwinden. Ist dieses nicht der Fall, so berechne $syz(f_2, \frac{\partial f_2}{\partial y})$ und erhalte, wie in 3.2.1 beschrieben, die Koordinatentransformation

$$\varphi_2 : \begin{cases} x \mapsto x \\ y \mapsto y - \frac{b}{a} z \\ z \mapsto z \end{cases}$$

Es gilt nun: $j_3(\varphi_2 \circ \varphi_1 \circ f) = xy^2 + f_3(y,z)$

Im dritten Teilschritt gilt es, f_3 zu transformieren. Wende dazu die Tschirnhaustransformation auf $f_3(y,z) = az^3 + byz^2 + \ldots$ an:

$$\varphi_3 : \begin{cases} x \mapsto x \\ y \mapsto y \\ z \mapsto z - \frac{b}{3a} y \end{cases}$$

Es können dabei aber Terme der Form y^3 und $y^2 z$ entstehen. Daher folgt:
$g(x,y,z) := j_3(\varphi_3 \circ \varphi_2 \circ \varphi_1 \circ f) = f_1(x,y,z)y^2 + z^3$

Berechne dazu $f_1(x,y,z) = (g(x,y,z) - z^3)/y^2$ mit $g(x,y,z) = ax + by + cz$. Damit haben wir direkt die Transformation

$$\varphi_4 : \begin{cases} x \mapsto x - \frac{b}{a} y - \frac{c}{a} z \\ y \mapsto y \\ z \mapsto z \end{cases}$$

und es gilt:
$j_3(\varphi_4 \circ \varphi_3 \circ \varphi_2 \circ \varphi_1 \circ f) = \varepsilon x y^2 + \varepsilon' z^3$

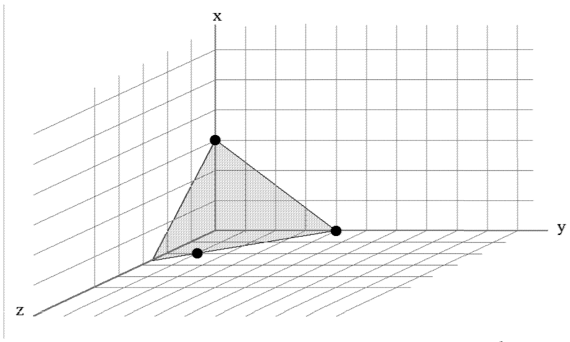
Zum Abschluß bleibt nur noch, die Koordinaten adäquat zu vertauschen, um die Form $x^3 + yz^2$ zu erhalten.

Jetzt kann es mit dem nächsten Schritt weiter gehen.

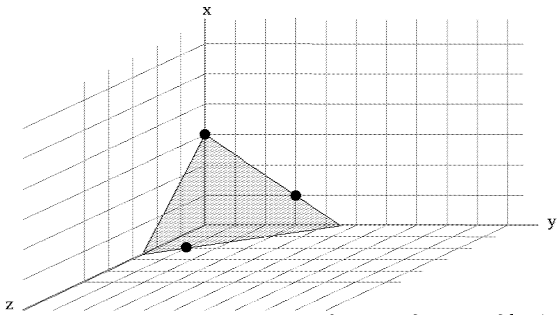
3.3.2.1 Schritt 59

In diesem Schritt werden über das 3-dimensionale Newtondiagramm vier Möglichkeiten unterschieden.
Das Prinzip ist wie im 2-Dimensionalen, nur daß wir hier zwei Punkte festhalten und um diese zwei Punkte die gesamte Ebene drehen; dh. in diesem Fall sind x^3 und yz^2 fest und wir ziehen die Spitze auf der y-Achse immer weiter nach rechts. Haltepunkte sind auch hier wieder diejenigen Monome, die von der Ebene getroffen werden.

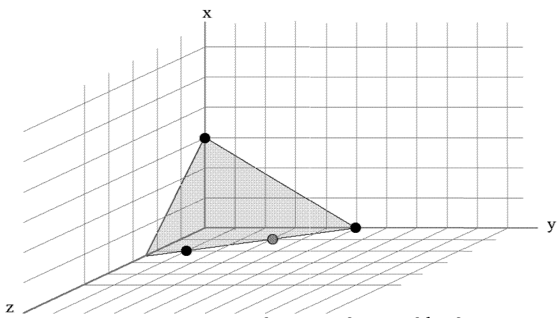
Normalform	Typ	Schritt
$j_{yz^2,x^3,y^{3k+1}}f \approx\ x^3+yz^2+y^{3k+1}$	Q_{6k+4}	60_k
$j_{yz^2,x^3,xy^{2k+1}}f \approx\ x^3+yz^2+xy^{2k+1}$	Q_{6k+5}	61_k
$j_{yz^2,x^3,y^{3k+2}}f \approx\ x^3+yz^2+y^{3k+2}$	Q_{6k+6}	62_k
$j_{yz^2,x^3,y^{3k+2}}f =\ \varepsilon x^3+\varepsilon' yz^2$		63_{k+1}



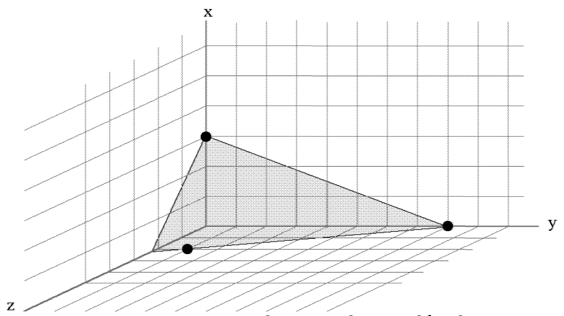
Fall: $j_{yz^2,x^3,y^{3k+1}}f \approx\ x^3+yz^2+y^{3k+1}$



Fall: $j_{yz^2,x^3,xy^{2k+1}}f \approx\ x^3+yz^2+xy^{2k+1}$



Fall: $j_{yz^2,x^3,y^{3k+2}}f \approx\ x^3+yz^2+y^{3k+2}$ für $k=1$



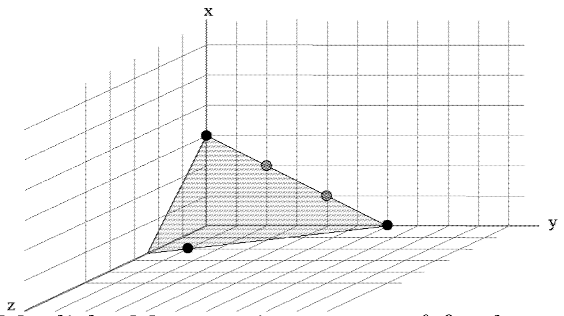
Fall: $j_{yz^2,x^3,y^{3k+2}}f \approx\ x^3 + yz^2 + y^{3k+2}$ für $k = 2$

Auch hier genügt es wieder, nur die Existenz des Monomes zu überprüfen. Der vierte Fall zerfällt in drei Gruppen, die mit Hilfe von Dimension und Multiplizität unterschieden werden.

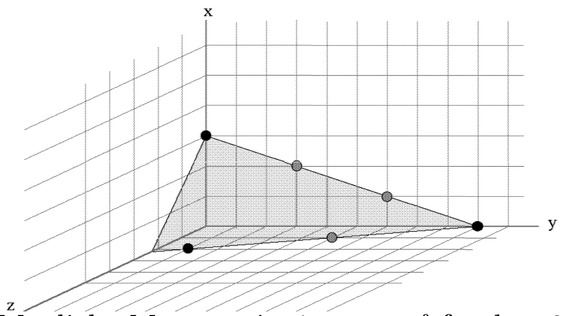
Normalform	Typ	d	m	Schritt
$j_{yz^2,x^3,y^{3k}}f \approx\ x^3 + yz^2 + ax^2y^k + xy^{2k}\ ,\ a^2 \neq 4$	$Q_{k,0}$	0		64_k
$j_{yz^2,x^3,y^{3k}}f \approx\ x^3 + yz^2 + x^2y^k$	$Q_{k,p}$	1	1	65_k
$j_{yz^2,x^3,y^{3k}}f \approx\ x^3 + yz^2$		1	2	59_k

wobei $d = dim(\mathbb{C}\{x, y, z\}/J_f) = \mathtt{dim}(\mathtt{std}(Jf))$ und m gleich der Multiplizität des lokalen Ringes $\mathbb{C}\{x, y, z\}/J_f$ ist d.h $m = \mathtt{mult}(\mathtt{std}(Jf))$.

(Wie das Beispiel $f = (x + y^2)^3 + yz^2 + y^7$ zeigt, gilt in der letzten Zeile der Tabelle "$\approx$" statt "$=$".)



Mögliche Monome in $j_{yz^2,x^3,y^{3k}}f$ für $k = 2$



Mögliche Monome in $j_{yz^2,x^3,y^{3k}}f$ für $k = 3$

Mit der dritten Zeile beginnen wir wieder bei Schritt 59_k.

3.3.3 Schritt 66

$f \sim \varphi + \alpha(y) + x\beta(y) + z\gamma(y)$ mit $\varphi = x^2z + yz^2$ und $j_3(\alpha, x\beta, z\gamma) = 0$. Ich habe leider noch keinen Weg gefunden dem Computer hier die nötige menschliche Intuition beizubringen, um die passende Koordinatentransformation zu finden. Denn als erstes muß ersteinmal die Rolle von x, y und z gefunden werden, bevor man auf die Suche nach der Transformation gehen kann.
Dieser Fall ist derzeit noch nicht implementiert.

3.3.4 Schritt 82

Wie immer muß eine passende Koordinatentransformation gefunden werden, so daß $j_3 f = x^3 + xz^2$ gilt.

Führe dazu zunächst eine Koordinatentransformation durch, so daß f danach nur noch von zwei Veränderlichen abhängt. (Seien x und z diese.)
Sei φ diese Koordinatentransformation. Bezeichne mit $\underline{x} = (x, y, z) = (x_1, x_2, x_3)$. und $\underline{x}' = (x'_1, x'_2, x'_3)$

Wir haben also $\underline{x} = \varphi(\underline{x}') = \begin{pmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{pmatrix} \begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix} =: A\underline{x}'$

Da $(f \circ \varphi)(\underline{x}')$ unabhängig von x'_2 ist, gilt $\frac{\partial}{\partial x'_2}(f \circ \varphi)(\underline{x}') \equiv 0$

Nach der Kettenregel ergibt sich $[\frac{\partial}{\partial x'_2}(f \circ \varphi)]|_{\underline{x}'} = \sum_{i=1}^{3} \frac{\partial f}{\partial x_i}(\varphi(\underline{x}')) \cdot \frac{\partial \varphi_i}{\partial x'_2}(\underline{x}') \equiv 0$

Anwenden von $\varphi^{-1}(\underline{x})$ liefert $\sum_{i=1}^{3} \frac{\partial f}{\partial x_i}(\underline{x}) \cdot b_i \equiv 0$, wobei $b_i := \frac{\partial \varphi_i}{\partial x'_2}(\underline{x}')$ ist.

d.h. $(\, f_x, f_y, f_z\,) \cdot \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = 0$

$\Rightarrow \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \texttt{syz}(\texttt{jacob}(f))[1 \ldots 3, 1]$

Wir erhalten somit $\begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} \neq 0$, wobei $\frac{\partial(f \circ \varphi)}{\partial x'_i} \equiv 0$ aus $b_i \neq 0$ folgt. Es gilt also

- für $i = 1$:
  $A = \begin{pmatrix} b_1 & 0 & 0 \\ b_2 & 1 & 0 \\ b_3 & 0 & 1 \end{pmatrix}$
- für $i = 2$:
  $A = \begin{pmatrix} 1 & b_1 & 0 \\ 0 & b_2 & 0 \\ 0 & b_3 & 1 \end{pmatrix}$
- für $i = 3$:
  $A = \begin{pmatrix} 1 & 0 & b_1 \\ 0 & 1 & b_2 \\ 0 & 0 & b_3 \end{pmatrix}$

Das Ergebnis ist nun ein homogenes Polynom vom Grad 3 in 2 Variablen. d.h $j_3f = ax^3 + bx^2z + cxz^2 + dz^3$. Es gilt also die geeignete Koordinatentransformation zu finden, so daß $j_3f = x^3 + xz^2$ ist. Betrachte dazu 3 Fälle:

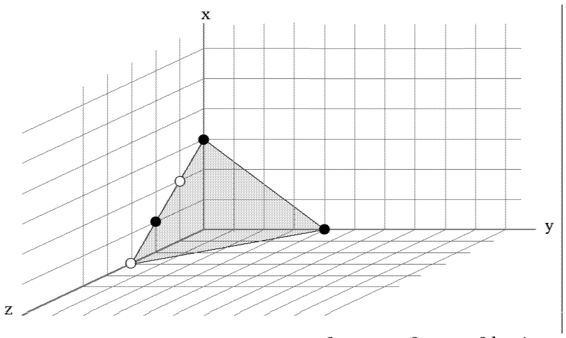
1. $a = d = 0$ so gilt $f = xz(bx + cz)$ wobei $b \neq 0$ und $c \neq 0$ sind, da sonst Fall 8 (Schritt 97) aus dem Satz zutreffen würde.
   Die Transformation $z \mapsto z - \frac{bx}{2c}$ liefert $j_3f = x^3 + xz^2$
2. O.B.d.A sei $a \neq 0$ und $d = 0$ Die Tschirnhaustransformation $x \mapsto x - \frac{bz}{3}$ liefert $j_3f = x^3 + xz^2$
3. $a \neq 0$ und $d \neq 0$ (Körpererweiterung)
   Hier endet die Implementierung dieses Teiles.

Für die Fälle in denen die Transformation erfolgreich durchgeführt werden konnte, geht es mit Schritt 83_1 weiter.

3.3.4.1 Schritt 83_k

Zunächst einmal die Fallunterscheidung mit dem Newtondiagramm.

Normalform	Typ	Schritt
$j_{x^3,z^3,y^{3k+1}}f \approx \ x^3 + xz^2 + y^{3k+1}$	U_{12k}	84_k
$j_{x^3,z^3,y^{3k+1}}f = \ x^3 + xz^2$		85_k

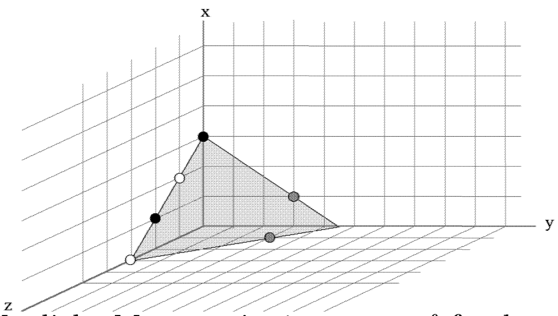


Fall: $j_{x^3,z^3,y^{3k+1}}f \approx \ x^3 + xz^2 + y^{3k+1}$

Gefolgt von einer Auswertung von Dimension und Multiplizität.

Normalform	Typ	d	Schritt
$j_{x^3,z^3,xy^{2k+1}}f \approx \ x^3 + xz^2 + xy^{2k+1} + czy^{2k+1}$	$U_{k,0}$	0	86_k
$j_{x^3,z^3,xy^{2k+1}}f \approx \ x^3 + xz^2 + xy^{2k+1}$	$U_{k,p}$	1	87_k
$j_{x^3,z^3,xy^{2k+1}}f = \ x^3 + xz^2$			88_k

wobei $d = dim(\mathbb{C}\{x,y,z\}/J_f) = \mathtt{dim}(\mathtt{std}(Jf))$.

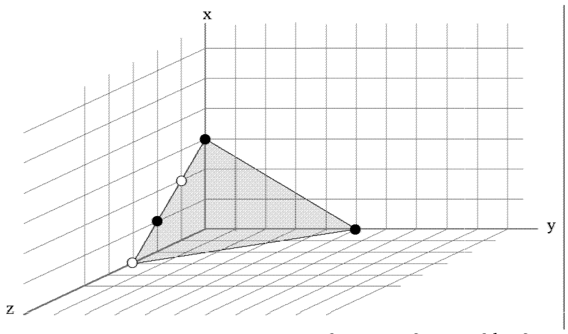


Mögliche Monome in $j_{x^3,z^3,xy^{2k+1}}f$ für $k=1$

3.3.4.2 Schritt 88_k

Wieder eine Existenzprüfung.

Normalform	Typ	Schritt
$j_{x^3,z^3,y^{3k+2}}f \approx\ x^3+xz^2+y^{3k+2}$	U_{12k+4}	89_k
$j_{x^3,z^3,y^{3k+2}}f =\ x^3+xz^2$		90_{k+1}



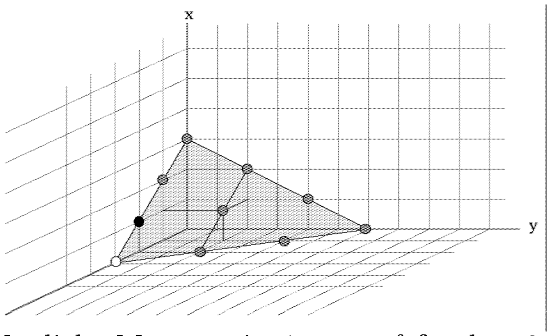
Fall: $j_{x^3,z^3,y^{3k+2}}f \approx\ x^3+xz^2+y^{3k+2}$

3.3.4.3 Schritt 90_k

Da in diesem Schritt die Klassifikation von Arnold eine Lücke aufweist, ist hier nur der Schleifeschluß implementiert. Z.B wird die Singularität $f=x^3+xz^2+y^6$ in der folgenden Tabelle nicht erfaßt, denn $dim(\mathbb{C}\{x,y,z\}/J_f)=0$.

Normalform	Typ	d	m	Schritt
$j_{x^3,z^3,y^{3k}}f \approx\ x^2z + xz^2 + ax^2y^k + bxy^kz + y^kz^2 + cxy^{2k}$	$U^*_{k,0}$	1	2	91_k
$j_{x^3,z^3,y^{3k}}f \approx\ x^2z + xz^2 + ax^2y^k + bxy^kz + y^kz^2$	UP_k	1	1	92_k
$j_{x^3,z^3,y^{3k}}f \approx\ x^3 + xz^2 + ax^2z + y^kz^2$	UQ_k	1	2	93_k
$j_{x^3,z^3,y^{3k}}f \approx\ x^2z + xz^2 + x^2y^k + axy^kz$	UR_k	1	3	94_k
$j_{x^3,z^3,y^{3k}}f \approx\ x^2z + xz^2 + x^2y^k$	US_k	1	3	95_k
$j_{x^3,z^3,y^{3k}}f \approx\ x^2z + xz^2 + xy^kz$	UT_k	1	3	96_k
$j_{x^3,z^3,y^{3k}}f \approx\ x^2z + xz^2$		1	4	83_k

wobei $d = dim(\mathbb{C}\{x,y,z\}/J_f) = \mathtt{dim}(\mathtt{std}(Jf))$ und m gleich der Multiplizität des lokalen Ringes $\mathbb{C}\{x,y,z\}/J_f$ ist d.h $m = \mathtt{mult}(\mathtt{std}(Jf))$.



Mögliche Monome in $j_{x^3,z^3,y^{3k}}f$ für $k = 2$.

In jedem Fall bildet diese Tabelle das Ende der Schleife und fährt mit Schritt 83_k fort.

Zuerst muß jedoch erst einmal eine passende Koordinatentransformation gefunden werden so, daß $j_{x^3,z^3,y^{3k}}f = x^3 + xz^2$.

Wie sieht diese aus?
Setze dazu allgemein folgende Form an:

$$j_{x^3,z^3,y^{3k}}f = (ax + by^k)^3 + (ax + by^k) \cdot (b'y^k + c'z)^2$$

Dies ist möglich, da jeder neue Term y^k als Faktor enthält, und der Anteil $x^3 + xz^2$ nicht verändert werden muß.
Es bleiben nun 4 Möglichkeiten zu untersuchen:
Betrachte dazu die jeweiligen Koeffizienten der Monome von $j_{x^3,z^3,y^{3k}}f$

	$b=0$ $b'=0$	$b=0$ $b'\neq 0$	$b\neq 0$ $b'=0$	$b\neq 0$ $b'\neq 0$
x^3	a^3	a^3	a^3	a^3
x^2y^k	0	0	$3a^2b$	$3a^2b$
xy^{2k}	0	ab'^2	$3ab^2$	$3ab^2+ab'^2$
y^{3k}	0	0	b^3	$b^3+bb'^2$
xz^2	ac'^2	ac'^2	ac'^2	ac'^2
z^2y^k	0	0	bc'^2	bc'^2
zy^{2k}	0	0	0	$2bb'c'$
xy^kz	0	$2ab'c'$	0	$2ab'c'$

- $b=b'=0 \Rightarrow j_{x^3,z^3,y^{3k}}f = j_{x^3,z^3,y^{3k-1}}f$

- $b=0$ und $b'\neq 0$
  Prüfe dazu ob, der Koeffizient von (x^2y^k) verschwindet.
  Setze $g = (j_{x^3,z^3,y^{3k}}f - a^3x^3)/x$. Dann liefert $syz(g, \frac{\partial g}{\partial z})$ die Koordinatentransformation

$$\varphi : \begin{cases} x \mapsto x \\ y \mapsto y \\ z \mapsto z - \frac{b'}{c'}y^k \end{cases}$$

  φ wird in der Prozedur Isomorphie_s82_z() bestimmt und durchgeführt.

- $b\neq 0$ und $b'=0$
  Prüfe dazu ob, der Koeffizient von xy^kz verschwindet.
  Setze $g = j_{x^3,z^3,y^{3k}}f - ac'^2xz^2 - bc'^2z^2y^k$. Dann liefert $syz(g, \frac{\partial g}{\partial x})$ die Koordinatentransformation.

$$\varphi : \begin{cases} x \mapsto x - \frac{b}{a}y^k \\ y \mapsto y \\ z \mapsto z \end{cases}$$

  φ wird in der Prozedur Isomorphie_s82_x() bestimmt und durchgeführt.

- $b\neq 0$ und $b'\neq 0$
  Sei zunächst $\varphi : \begin{cases} x \mapsto x - \frac{2bb'c'}{2ab'c'}y \\ y \mapsto y \\ z \mapsto z \end{cases}$.
  Dann hat $j_{x^3,z^3,y^{3k}}(\varphi(f))$ die Form von Fall 2. Wende diesen an.
  D.h. starte die Prozedur Isomorphie_s82_z() für $\varphi(f)$.

3.3.5 Schritt 97

Wegen $f \approx x^2y$ gilt $f = l_1^2 l_2$ wobei $l_1 = ax + by + cz$, $l_2 = a'x + b'y + c'z$ Linearformen sind.
O.B.d.A können wir in dieser Erläuterung annehmen, daß $\frac{\partial^2 f}{\partial x^2} \neq 0$ ist.

Wir erhalten: $\frac{\partial f}{\partial x} = l_1(\frac{\partial(l_1 l_2)}{\partial x} + \frac{\partial l_1}{\partial x} l_2)$
Setze $h := \frac{\partial(l_1 l_2)}{\partial x} + \frac{\partial l_1}{\partial x} l_2$
Wegen $a \neq 0$ erhalten wir $(f, \frac{\partial f}{\partial x}) \begin{pmatrix} h \\ -l_1 l_2 \end{pmatrix} = 0$
In Singular: $-l_1 l_2 =$`syz`$(f, \frac{\partial f}{\partial x})$`[2,1]`
Da wir nun $l_1 l_2$ kennen, läßt sich l_1 wie folgt berechnen: $(f,$`syz`$(f, \frac{\partial f}{\partial x})$`[2,1]`$) \begin{pmatrix} 1 \\ l_1 \end{pmatrix} =$ 0, d.h. $l_1 =$`syz`$(f,$`syz`$(f, \frac{\partial f}{\partial x})$`[2,1]`$)$
Die Koordinatentransformation $\varphi_1 : \begin{cases} x \mapsto x - by - cz \\ y \mapsto y \\ z \mapsto z \end{cases}$ liefert uns $f \circ \varphi_1 = x^2 l_2$, wobei $l_2 = \alpha x + \beta y + \gamma z$ linear, mit $(\beta, \gamma) \neq (0,0)$ ist.
Somit gilt $l_2 = \frac{f}{x^2}$

Sei O.B.d.A $\beta \neq 0$, sonst vertausche y und z.
Setze $\varphi_2 : \begin{cases} x \mapsto x \\ y \mapsto \frac{1}{\beta}(-\alpha x + y - \gamma z) \\ z \mapsto z \end{cases}$ und erhalte damit $j_3 f \circ \varphi_1 \circ \varphi_2 = x^2 y$.

3.3.5.1 Schritt 98

Nachdem nun die Koordinatentransformation bestimmt und durchgeführt wurde, geht es mit Hilfe von Dimension und Multiplizität weiter.

Normalform	d	m	Typ	Schritt
$j_{x^2y,y^4,z^4} f \approx x^2y + z^4 + az^3y + bz^2y^2 + zy^3$, $\Delta \neq 0$, $ab \neq 9$	0		$V_{1,0}$	99
$j_{x^2y,y^4,z^4} f \approx x^2y + z^4 + bz^3y + z^2y^2$, $b^2 \neq 4$	1	1	$V_{1,p}$	100
$j_{x^2y,y^4,z^4} f \approx x^2y + z^3y + az^2y^2 + y^4$, $4a^3 + 27 \neq 0$	1	2	$V_{1,p}^{\#}$	101
$j_{x^2y,y^4,z^4} f \approx x^2y + \varphi$				102

wobei das φ eines der folgenden sein kann:

1. $f \sim x^2y + z^4 + z^3y + z^2y^2 + zy^3$
2. $f \sim x^2y + z^4 + z^3y + z^2y^2$
3. $f \sim x^2y + z^3y + z^2y^2 + y^4$
4. $f \sim x^2y + z^4 + z^3y$

5. $f \sim x^2y + z^3y + z^2y^2$
6. $f \sim x^2y + z^2y^2 + zy^3$
7. $f \sim x^2y + z^4 + z^2y^2$
8. $f \sim x^2y + z^4$
9. $f \sim x^2y + z^3y$
10. $f \sim x^2y + z^2y^2$
11. $f \sim x^2y + zy^3$
12. $f \sim x^2y + y^4$
13. $f \sim x^2y$

3.3.6 Schritte 103 bis 105

Damit endet der Algorithmus von Arnold, da in Schritt 103 bis 105 keine Bestimmung von Normalformen mehr erfolgt. Es werden nur noch Abschätzungen der Milnorzahl, der Modalität und des corangs abgegeben.

Anhang A

Klassifikation mittels "Milnorcode"

Im Laufe der Implementierung des Algorithmus hat sich zeigt, daß man mit Hilfe des Hilbert-Polynoms der Milnor-Algebra die Hyperflächensingularitäten in einem gewissen Rahmen unterscheiden kann.

In diesem Anhang sind in tabellarischer Form die Ergebnisse der Tests für generische Moduli zusammengefaßt. Für spezielle Moduli kann der Milnorcode abweichend sein. Zum Beispiel hat $J[k,0]$ die allgemeine Form:
$x^3 + bx^2y^k + y^{3k} + cxy^{2k+1}$ mit $4b^3 + 27 \neq 0$, $k > 1$ und
$c = c_0 + \cdots + c_{k-3}y^{k-3}$, für k=2 ist $c = 0$.

Für den speziellen Modulus $b = 0$ ändert sich jedoch der Milnorcode, z.B. ergibt sich für $J[4,0]$:

$$\begin{array}{lll}
b=2,\ c=0 & : x^3 + 2x^2y^4 + xy^9 + y^{12} & 1,7,7 \\
b=2,\ c=1+y & : x^3 + 2x^2y^4 + xy^9 + xy^{10} + y^{12} & 1,7,7 \\
b=0,\ c=0 & : x^3 + xy^9 + y^{12} & 1,8,5 \\
b=0,\ c=1+y & : x^3 + xy^9 + xy^{10} + y^{12} & 1,8,5
\end{array}$$

(Implementiert ist nur der Milnorcodes für generische Moduli).

Zum Anderen können verschieden Singularitäten den selben Milnorcode haben: z.B. Z$[k,r,s]$) und Y$[k,2r,s+4r]$) oder auch Z_r^1 und $Y_{3,s}^1$.

Trotzdem kann man mit dieser Methode sehr schnell und mit hoher Wahrscheinlichkeit den Typ der Singularität für generische Moduli bestimmen.

Das Hilbert-Polynom der Minor-Algebra $K[x_1, \ldots, x_n]/J_f$ ist ein Polynom $P(t) = \sum_{i=0}^{\nu} a_i \cdot t^i$, $a_i = dim_{\mathbb{C}}((\mathcal{M}^i + J_f)/(\mathcal{M}^{i+1} + J_f))$. Hierbei gibt es einen größten Index $\rho \geq 1$ und einen kleinsten Index ρ', so daß für $i < \rho$ die Folge der (a_i) monoton steigend, und für $i > \rho'$ monoton fallend ist. Zwischen ρ und ρ' ist die Folge konstant.

Notiert man nun wie oft jeder Koeffizient a_i vorkommt erhält man einen $2 \cdot a_\rho - 1 \leq \nu + 1$ langen Vektor $(c_1, \ldots, c_{2a_\rho+1})$. Diese Vektoren sind, nach längeren Versuchsreihen in den folgenden Tabellen aufgetragen.

z.B. hat A$[k]$ den Milnorcode (k), also das Hilbertpolynom
$P(t) = 1 + t + t^2 + \cdots + t^{k-1}$
und W$[12k]$ den Milnorcode $(1, 1, 2k, 2k-1, 2k-1)$ also das Hilbertpolynom
$P(t) = 1 + 2t + 3t^2 + \cdots + 3t^{2k+1} + 2t^{2k+2} + \cdots + 2t^{4k} + t^{4k+1} + \cdots + t^{6k-1}$.

Für alle Tabellen gilt $k, r, s \geq 0$.

Corang 1

	Bedingung	Milnorcode	Milnorzahl	Modalität
		c_1	μ	M
$A[k]$	$k \geq 0$	k	k	0

Corang 2 und $Jet^3 f \neq 0$.

	Bedingung	Milnorcode			Milnorzahl	Modalität
		c_1 1	c_2 2	c_3 1	μ	M
$D[k]$	$k \geq 4$	1	1	k-3	k	0
$J[k,r]$	$k \geq 2$	1	2k-1	2k-1+r	6k-2+r	k-1
$E[6k]$	$k \geq 1$	1	2k	2k-1	6k	k-1
$E[6k+1]$	$k \geq 1$	1	2k	2k	6k+1	k-1
$E[6k+2]$	$k \geq 1$	1	2k+1	2k-1	6k+2	k-1

Tabelle für Corang 2 und $Jet^4 f \neq 0$

	Bed.	Milnorcode					Milnorzahl	Modalität
		c_1	c_2	c_3	c_4	c_5	μ	M
		1	2	3	2	1		
$W[12k]$		1	1	2k	2k-1	2k-1	12k	3k-2
$W[12k+1]$		1	1	2k	2k-1	2k	12k+1	3k-2
$W[12k+5]$		1	1	2k+1	2k-1	2k+1	12k+5	3k-1
$W[12k+6]$		1	1	2k+1	2k	2k	12k+6	3k-1
$W[k,r]$	$r \geq 0$	1	1	2k	2k	2k+r	12k+3+r	3k-1
$W*[k,2r-1]$	$r > 0$	1	1	2k	2k+r	2k-1	12k+2+2r	3k-1
$W*[k,2r]$	$r > 0$	1	1	2k	2k+r	2k	12k+3+2r	3k-1
$X[k,r]$	$k \geq 1$	1	1	2k-1	2k-1	2k-1+r	12k-3+r	3k-2
$Y[k,r,s]$	$k \geq 1$, $r \leq s$	1	1	2k-1	2k-1+r	2k-1+r-s	12k+r+s-3	3k-2
$Z[1,r]$	$r \geq 0$	1	1	1	3	r+3	15+r	3k+r-2
$Z[k,r,s]$	$r > 0$	1	1	2k-1	2k-1+2r	2k-1+2r+s	12k+6r+s-3	3k+r-2
$Z[k,12k+6r-1]$		1	1	2k-1	2k+2r	2k+2r-1	12k+6r-1	3k+r-2
$Z[k,12k+6r]$		1	1	2k-1	2k+2r	2k+2r	12k+6r	3k+r-2
$Z[k,12k+6r+1]$		1	1	2k-1	2k+2r+1	2k+2r-1	12k+6r+1	3k+r-2

$h(Y^k_{r,s}) = h(Z^k_{r/2,s-2r})$ bzw. $h(Z^k_{r,s}) = h(Y^k_{2r,s+4r})$
$h(Z^1_r) = H(Y^1_{3,s})$ aber nur wenn $c_5 \geq c_4$

Corang 3

	Bedingung	Milnorcode					Milnorzahl	Modalität
		c_1	c_2	c_3	c_4	c_5	μ	M
		1	2	3	2	1		
$Q[k,0]$	$k>1$	1	0	2	2k-2	2k-1	6k+2	k
$Q[k,r]$	$k>1$	1	0	2	2k-2	2k-1+r	6k+2+r	k
$Q[6k+4]$		1	0	2	2k-1	2k-1	6k+4	k
$Q[6k+5]$		1	0	2	2k-1	2k	6k+5	k
$Q[6k+6]$		1	0	2	2k	2k-1	6k+6	k
$S[12k-1]$		1	0	2k+1	2k-2	2k-1	12k-1	3k-2
$S[12k]$		1	0	2k+1	2k-2	2k	12k	3k-2
$S[k,0]$		1	0	2k+1	2k-1	2k	2k+2	3k-1
$S[k,r]$	$r>0$	1	0	2k+1	2k-1	2k+r	12k+2+r	3k-1
$S*[k,2r-1]$	$r>0$	1	0	2k+1	2k-1+r	2k-1	12k+1+2r	3k-1
$S*[k,2r]$	$r>0$	1	0	2k+1	2k-1+r	2k	12k+2+2r	3k-1
$S[12k+4]$		1	0	2k+2	2k-2	2k+1	12k+4	3k-1
$S[12k+5]$		1	0	2k+2	2k-1	2k	12k+5	3k-1
$T[p,q,r]$	$p \leq q \leq r$	1	0	p-1	q-p	r-q+1	p+q+r-1	

	Bedingung	Milnorcode							Milnorzahl	Modalität
		c_1	c_2	c_3	c_4	c_5	c_6	c_7	μ	M
		1	2	3	4	3	2	1		
$U[12k]$	$k \geq 1$	1	0	1	k	2k-1	0	2k-1	12k	4k-3
$U[12k+4]$	$k \geq 1$	1	0	1	k	2k	0	2k	12k+4	4k-2
$U[k,2r]$	$k \geq 1$	1	0	1	k	2k-1	r+1	2k-1	12k+2r+2	4k-2
$U[k,2r-1]$	$k \geq 1$	1	0	1	k	2k-1	r	2k	12k+2r+1	4k-2
$V[1,0]$		1	0	1	2	0	1	1	15	3
$V[1,r]$	$r>0$	1	0	1	2	0	1	r+1	15+r	3
$V*[1,2r-1]$	$r>0$	1	0	1	2	1	r-1	1	2r+14	3
$V*[1,2r]$	$r>0$	1	0	1	2	1	r-1	2	2r+15	3

Anhang B

Funktionen

Die Implementierung des Klassifizierers besteht im wesentlichen aus einer Library von ca 90 kBytes (entspricht ca 44 Seiten). Die Library classify.lib besteht aus 67 Prozeduren, davon stehen den Nutzer allerdings nur 15 zur Verfügung (s.u.). Die anderen sind internen Zwecken vorbehalten. Die zusätzliche Library makedbm.lib wird nur vor der allerersten Nutzung des Klassifizierers benötigt, um die Datenbank der Singularitäten zu erzeugen. In dieser Datenbank (im Unix DBM-format[1] werden zu jeder Singularität die Normalform, Corang, Milnorzahl und Milnorcode abgespeichert. Die Datenbankdateien heißen NFlist.dir und NFlist.pag. Sie werden mit der Prozedur create_sing_dbm() erzeugt. read_sing_dbm() bietet die Möglichkeit, Daten wieder auszulesen. Zum jetzigen Zeitpunkt sind 55 Klassen von Singularitäten abgespeichert.

Auf Grund des Umfangs ist hier nicht die gesamte Library wiedergegeben, sondern nur der Hilfetext der Prozeduren, die dem Nutzer auch zugänlich sind. Die vollständige Version des Klassifizierers liegt auf Diskette bei oder ist auch unter URL=http://www.mathematik.uni-kl.de/~krueger/Singular/ zu finden.

Zur Nutzung des Klassifizierers genügt es die Library classify.lib, mit Hilfe des Singularbefehls LIB"classify.lib";, zu laden:

Es stehen dem Nuzter folgende Funktionen zur Verfügung (zu jeder Funktion werden der Hilfstext und ein Beispiel angegeben):

[1] dieses ist in Singular direkt implementiert und damit auch unter Microsoft Betriebssystemen und Macintosh verfügbar (Version 1.1 oder höher).

```
> help classify;
// proc classify from lib classify.lib
USAGE:    classify(f);  f=poly
COMPUTE:  normal form and singularity type of f with respect to right
          equivalence, as given in the book "Singularities of
          differentiables maps, Volume I" by V.I. Arnold, S.M. Gusein-Zade,
          A.N. Varchenko
RETURN:   normal form of f, of type poly
REMARK:   This version of classify is only alpha. Please send bugs and
          comments to: "Kai Krueger" <krueger@mathematik.uni-kl.de>
          Be sure to have at least Singular version 0.9.3, better 1.0.1
          Updates can be found under:
          URL=http://www.mathematik.uni-kl.de/~krueger/Singular/
NOTE:     type init_debug(n); (0 <= n <= 10) in order to get intermediate
          information, higher values of n give more information.
          The proc creates several global objects with names all starting
          with @, hence there should be no name conflicts
EXAMPLE:  example classify; shows an example
> example classify;
// proc classify from lib classify.lib
EXAMPLE
   ring r=0,(x,y,z),ds;
   poly f=(x2+3y-2z)^2+xyz-(x-y3+x2*z3)^3;
   classify(f);
About the singularity :
          Milnor number(f)   = 4
          Corank(f)          = 2
          Determinacy       <= 5
Guessing type via Milnorcode:
 D[k]=D[4]

Computing normal form ...
I have to apply the splitting lemma. This will take some time....:-)
   Arnold step number 4
The singularity
   `-x3+3/2xy2+1/2x3y-1/16x2y2+3x2y3'
is R-equivalent to D[4].
   Milnor number = 4
   modality      = 0
z2+x2y+y3
   init_debug(3);
```

```
Debugging level change from  0  to  3
   classify(f);
Computing Basicinvariants of f ...
About the singularity :
          Milnor number(f)   = 4
          Corank(f)          = 2
          Determinacy       <= 5
Hcode:
1,2,1,0,0
Milnor code :
1,1,1
Debug:(2):
entering HKclass3_teil_1
1,1,1
Debug:(2):
finishing HKclass3_teil_1
Guessing type via Milnorcode:
 D[k]=D[4]

Computing normal form ...
I have to apply the splitting lemma. This will take some time....:-)
Debug:(3):
Spalte folgendes Polynom mit Bestimmtheit:
5
Debug:(3):
9y2-12yz+4z2-x3+6x2y-4x2z+xyz+x4+3x2y3
Debug:(2):
Reihenfolge fuer Vertauschungen:
3,2,1
Debug:(2):
Reihenfolge fuer Vertauschungen:
3,2,1
Debug:(2):
rank determined with Morse rg=
1
Rest singularity f=
-x3+3/2xy2+1/2x3y-1/16x2y2+3x2y3
Step 3
   Arnold step number 4
The singularity
   `-x3+3/2xy2+1/2x3y-1/16x2y2+3x2y3'
```

```
is R-equivalent to D[4].
   Milnor number = 4
   modality      = 0
Debug:(2):
Decode:
Debug:(2):
S_in=
D[4]
  s_in=
D[4]
Debug:(2):
Looking for Normalform of
D[k]
with (k,r,s) = (
4
,
0
,
0
)
Debug:(2):
Opening Singalarity-database:


DBM: /home/staff/krueger/Diplom/Libs/NFlist
Debug:(2):
DBMread(
D[k]
)=
x2y+y^(k-1)
.
Debug:(2):
S=
f = x2y+y^(k-1);
 Tp=
x2y+y^(k-1)
Key=
I_D[k]
Polynom f=
x2y+y3
  crk=
```

```
2
  Mu=
4
 MlnCd=
1,1,1
Debug:(2):
Info=
x2y+y3
Debug:(2):
Normal form NF(f)=
x(3)^2+x(1)^2*x(2)+x(2)^3
z2+x2y+y3
```

```
> help init_debug;
// proc init_debug from lib classify.lib
USAGE:    init_debug([level]);  level=int
COMPUTE:  Set the global variable @DeBug to level. The variable @DeBug is
          used by the function debug_log(level, list of strings) to know
          when to print the list of strings. init_debug() reports only
          changes of @DeBug.
NOTE:     The procedure init_debug(n); is usefull as trace-mode. n may
          range from 0 to 10, higher values of n give more information.
EXAMPLE:  example init_debug; shows an example
> example init_debug;
// proc init_debug from lib classify.lib
EXAMPLE:
  init_debug();
  debug_log(1,"no trace information printed");
  init_debug(1);
Debugging level change from  0  to  1
  debug_log(1,"some trace information");
some trace information
  init_debug(2);
Debugging level change from  1  to  2
  debug_log(2,"nice for debugging scripts");
Debug:(2):
nice for debugging scripts
  init_debug(0);
Debugging switched off.
```

```
> help debug_log;
// proc debug_log from lib classify.lib
USAGE:    debug_log(level,li); level=int, li=comma separated "message" list
COMPUTE:  print "messages" if level>=@DeBug.
          useful for user-defined trace messages.
EXAMPLE:  example debug_log; shows an example
SEE ALSO: init_debug();
> example debug_log;
// proc debug_log from lib classify.lib
EXAMPLE:
  example init_debug;
// proc init_debug from lib classify.lib
EXAMPLE:
Debugging level change from  0  to  1
some trace information
Debugging level change from  1  to  2
Debug:(2):
nice for debugging scripts
Debugging switched off.
```

```
> help quickclass;
// proc quickclass from lib classify.lib
USAGE:    quickclass(f);         f=poly
RETURN:   Normal form of f in Arnold's list
REMARK:   try to determine the normal form of f by invariants, mainly by
          computing the Hilbert function of the Milnor albegra,
          no coordinate change is needed (see also proc 'milnorcode').
EXAMPLE:  example quickclass; shows an example
> example quickclass;
// proc quickclass from lib classify.lib
EXAMPLE:
   ring r=0,(x,y,z),ds;
   poly f=(x2+3y-2z)^2+xyz-(x-y3+x2*z3)^3;
   quickclass(f);
Singularity R-equivalent to :  D[k]=D[4]
normal form : x2y+y3
x2y+y3
```

```
> help swap;
// proc swap from lib classify.lib
USAGE:    swap(a,b);
RETURN:   b,a if b,a is the input (any type)
> example swap;
// proc swap from lib classify.lib
EXAMPLE:
  swap("variable1","variable2");
variable2 variable1
```

```
> help basicinvariants;
// proc basicinvariants from lib classify.lib
USAGE:    basicinvariants(f);   f = poly
COMPUTE:  Compute basic invariants of f: an upper bound d for the
          determinacy, the milnor number mu and the corank c of f
RETURN:   intvec: d, mu, c
EXAMPLE:  example basicinvariants; shows an example
> example basicinvariants;
// proc basicinvariants from lib classify.lib
EXAMPLE:
   ring r=0,(x,y,z),ds;
   basicinvariants((x2+3y-2z)^2+xyz-(x-y3+x2*z3)^3);
5,4,2
```

```
> help corank;
// proc corank from lib classify.lib
USAGE:    corank(f);   f=poly
RETURN:   the corank of the Hessian matrix of f, of type int
REMARK:   corank(f) is the number of variables occuring in the residual
          singulartity after applying 'morsesplit' to f
EXAMPLE:  example corank; shows an example
> example corank;
// proc corank from lib classify.lib
EXAMPLE:
  ring r=0,(x,y,z),ds;
  poly f=(x2+3y-2z)^2+xyz-(x-y3+x2*z3)^3;
  corank(f);
2
```

```
> help A_L;
// proc A_L from lib classify.lib
USAGE:    A_L(f);         f=poly
          A_L("name");    type=string
COMPUTE:  the normal form in Arnold's list of the singularity given either
          by a polynomial f or by its name.
RETURN:   A_L(f): compute via 'milnorcode' the class of f and
          return the normal form of f found in the database.
          A_L("name"): Get the normal form from the database for
          the singularity given by its name.
EXAMPLE:  example A_L; shows an example
> example A_L;
// proc A_L from lib classify.lib
EXAMPLE:
  ring r=0,(a,b,c),ds;
  poly f=A_L("E[13]");
  f;
a3+ab5+b8
  A_L(f);
The Milnor number of the function is infinite.
a3+ab5+b8
```

```
> help normalform;
// proc normalform from lib classify.lib
USAGE:    normalform(s);  s=string
RETURN:   Arnold's normal form of singularity with name s
EXAMPLE:  example normalform; shows an example.
> example normalform;
// proc normalform from lib classify.lib
EXAMPLE:
  ring r=0,(a,b,c),ds;
  normalform("E[13]");
a3+ab5+b8
```

```
> help singularity;
// proc singularity from lib classify.lib
USAGE:    singularity(t, l); t=string (name of singularity),
          l=list of integers (index/indices of singularity)
```

```
COMPUTE:  get the Singularity named by type t from the database.
          list l is as follows:
          l= k [,r [,s [,a [,b [,c [,d]]]]]] k,r,s=int   a,b,c,d=poly
          The name of the dbm-databasefile is: NFlist.[dir,pag]
          The file is found in the current directory. If it does not
          exists, please run the script MakeDBM first.
RETURN:   Normal form and corank of the singularity named by type t and its
          index (indices) l
EXAMPLE:  example singularity; shows an example
> example singularity;
// proc singularity from lib classify.lib
EXAMPLE
  ring r=0,(x,y,z),(c,ds);
  init_debug(0);
  singularity("E[6k]",6);
[1]:
   x3+xy13+y19
[2]:
   2
  singularity("T[k,r,s]", 3, 7, 5);
[1]:
   x3+xyz+z5+y7
[2]:
   3
  poly f=y;
  singularity("J[k,r]", 4, 0, 0, f);
[1]:
   x3+x2y4+y13
[2]:
   2
```

```
> help morsesplit;
// proc morsesplit from lib classify.lib
USAGE:    morsesplit(f);        f=poly
RETURN:   Normal form of f in M^3 after application of the splitting lemma
COMPUTE:  apply the splitting lemma (generalized Morse lemma) to f
EXAMPLE:  example morsesplit; shows an example
> example morsesplit;
// proc morsesplit from lib classify.lib
EXAMPLE
```

```
   ring r=0,(x,y,z),ds;
   export r;
   init_debug(1);
Debugging level change from  0  to  1
   poly f=(x2+3y-2z)^2+xyz-(x-y3+x2*z3)^3;
   poly g=morsesplit(f);
Rest singularity f=
-x3+3/2xy2+1/2x3y-1/16x2y2+3x2y3
   g;
-x3+3/2xy2+1/2x3y-1/16x2y2+3x2y3
```

```
> help Hcode;
// proc Hcode from lib classify.lib
USAGE:    Hcode(v); v=intvec
RETURN:   intvec, coding v according to the number of successive
          repetitions of an entry
EXAMPLE:  example Hcode; shows an example.
> example Hcode;
// proc Hcode from lib classify.lib
EXAMPLE:
  intvec v1 = 1,3,5,5,2;
  Hcode(v1);
1,0,1,0,2,0,0,1,0
  intvec v2 = 1,2,3,4,4,4,4,4,4,4,3,2,1;
  Hcode(v2);
1,1,1,7,1,1,1
```

```
> help milnorcode;
// proc milnorcode from lib classify.lib
USAGE:    milnorcode(f[,e]); f=poly, e=int
RETURN:   intvec, coding the Hilbert function of the e-th Milnor algebra
          of f, i.e. of basering/(jacob(f)^e) (default e=1), according
          to proc Hcode
EXAMPLE:  example milnorcode; shows an example
> example milnorcode;
// proc milnorcode from lib classify.lib
EXAMPLE:
  ring r=0,(x,y,z),ds;
```

```
  poly f=x2y+y3+z2;
  milnorcode(f);
1,1,1
  milnorcode(f,2);  // a big second argument may result in memory overflow
1,0,1,0,2,0,0,1,0
```

```
> help tschirnhaus;
// proc tschirnhaus from lib classify.lib
USAGE:    tschirnhaus();
> example tschirnhaus;
// proc tschirnhaus from lib classify.lib
```

```
> help internalfunctions;
// proc internalfunctions from lib classify.lib
USAGE:   internalfunctions();
RETURN:  nothing, display names of internal procedures of classify.lib
EXAMPLE: no example
> example internalfunctions;
// proc internalfunctions from lib classify.lib
```

```
> internalfunctions();
   Internal functions for the classification using Arnold's method,
   the function numbers correspond to numbers in Arnold's classifier:
Klassifiziere(poly f);            determine the type of the singularity f
  Funktion1bis (poly f, list cstn)
  Funktion3 (poly f, list cstn)
  Funktion6 (poly f, list cstn)
  Funktion13 (poly f, list cstn)
  Funktion17 (poly f, list cstn)
  Funktion25 (poly f, list cstn)
  Funktion40 (poly f, list cstn, int k)
  Funktion47 (poly f, list cstn)
  Funktion50 (poly f, list cstn)
  Funktion58 (poly fin, list cstn)
  Funktion59 (poly f, list cstn)
  Funktion66 (poly f, list cstn)
```

```
Funktion82 (poly f, list cstn)
Funktion83 (poly f, list cstn)
Funktion91 (poly f, list cstn, int k)
Funktion92 (poly f, list cstn, int k)
Funktion93 (poly f, list cstn, int k)
Funktion94 (poly f, list cstn, int k)
Funktion95 (poly f, list cstn, int k)
Funktion96 (poly f, list cstn, int k)
Funktion97 (poly f, list cstn)
Isomorphie_s82_x (poly f, poly fk, int k)
Isomorphie_s82_z (poly f, poly fk, int k)
Isomorphie_s17 (poly f, poly fk, int k, int ct)
printresult (string f, string typ, int Mu, int m, int corank, int K)

 Internal functions for the classifcation by invariants:
Cubic (poly f)
parity (int e)               return the parity of e
HKclass (intvec i)
HKclass3( intvec i, string SG_Typ, int cnt)
HKclass3_teil_1 (intvec i, string SG_Typ, int cnt)
HKclass5 (intvec i, string SG_Typ, int cnt)
HKclass5_teil_1 (intvec i, string SG_Typ, int cnt)
HKclass5_teil_2 (intvec i, string SG_Typ, int cnt)
HKclass7 (intvec i, string SG_Typ, int cnt)
HKclass7_teil_1 (intvec i, string SG_Typ, int cnt)

 Internal functions for the Morse-splitting lemma:
Morse(poly fi, int K, int corank)          Splittinglemma itself
Coeffs (list #)
Coeff

 Internal functions providing tools:
ReOrder(poly f)
Singularitaet (string typ,int k,int r,int s,poly a,poly b,poly c,poly d)
RandomPolyK
Faktorisiere(poly f, poly g, int p, int k)     compute g = (ax+by^k)^p
Teile(poly f, poly g);                   Teilt f durch g.
GetRf(poly f, int n);
Show(poly f);
checkring();
DecodeNormalFormString(string s);
```

```
  Setring(int n, string ringname);
```

B.1 Beispiel

```
LIB"classify.lib";
ring r=0,(x,y,z),ds;
poly f=A_L("J[6,3]")+z2;
// (classify.lib): Need to create database...
poly g=singularity("J[k,r]",6,3,0,(2-y+3y2)^2)[1]+z2;
f;
// z2+x3+x2y6+y21
g;
// z2+x3+x2y6+4y21-4y22+13y23-6y24+9y25
map phi=r,z,x,(x+y+z);
map psi=r,x,x+y2,z;

basicinvariants(f);
// 29,37,2

corank(f);
// 2

morsesplit(phi(f));
// z3+x6z2+x21
phi(f);
// x2+2xy+y2+2xz+2yz+z2+z3+x6z2+x21

classify(psi(f));
// About the singularity :
//          Milnor number(f)   = 76
//          Corank(f)          = 2
//          Determinacy       <= 56
// Guessing type via Milnorcode:
//  J[k,r]=J[12,6]
//
// Computing normal form ...
// I have to apply the splitting lemma. This will take some time....:-)
//    Arnold step number 12
```

```
// The singularity
// ‘x3+x8+6x7y2+15x6y4+20x5y6+15x4y8+6x3y10+x2y12+x21+21x20y2+210x19y4+
1330x18y6+5985x17y8+20349x16y10+54264x15y12+116280x14y14+203490x13y16+
293930x12y18+352716x11y20+352716x10y22+293930x9y24+203490x8y26+116280x7y28+
54264x6y30+20349x5y32+5985x4y34+1330x3y36+210x2y38+21xy40+y42’
// is R-equivalent to J[12,6].
//    Milnor number = 76
//    modality      = 11
// z2+x3+x2y12+y42

classify(phi(g));
// About the singularity :
//          Milnor number(f)   = 37
//          Corank(f)          = 2
//          Determinacy       <= 29
// Guessing type via Milnorcode:
//  J[k,r]=J[6,3]
//
// Computing normal form ...
// I have to apply the splitting lemma. This will take some time....:-)
//    Arnold step number 12
// The singularity
//    ‘x3+x2y6+4y21-4y22+13y23-6y24+9y25’
// is R-equivalent to J[6,3].
//    Milnor number = 37
//    modality      = 5
// z2+x3+x2y6+y21

quickclass(phi(f));
// Singularity R-equivalent to :  J[k,r]=J[6,3]
// normal form : x3+x2y6+y21
// x3+x2y6+y21

quickclass(phi(g));
// Singularity R-equivalent to :  J[k,r]=J[6,3]
// normal form : x3+x2y6+y21
// x3+x2y6+y21

milnorcode(phi(f));
// 1,11,14
```

Literaturverzeichnis

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

[Arn] Arnold,V.: Normal Forms for Functions near degenerate Critical Points..., FAA 6, S.254-272, 1972

[Ar2] Arnold,V.: Local Normal Forms of Functions, Inventiones mathematicae 35, 87-109, 1976

[Ati] M.F. Atiyah, I.G. MacDonald: Introduction to Commutive Algebra, Addison-Wesley, 1969

[Eis] D. Eisenbud: Commutive Algebra, Springer-Verlag, Graduate Texts in Mathematics 150, 1994

[Gat] K. Gatermann, R. Lauterbach: Automatic classification of normal forms, ZIB, Preprint SC 95-3, 1995

[GP] Greuel, G.-M.; G.Pfister: Advances and improvements in the theory of standard bases and syzygies. Arch. Math. **66**, 163– 1796 (1996).

[Har] J. Harris: Algebraic Geometry, Springer-Verlag, Graduate Texts in Mathematics 133, 1992

[Ha] R. Hartshorne: Algebraic Geometry, Springer-Verlag, Graduate Texts in Mathematics 52, 1993

[Laf] J.P. Lafon: Algebre commutative, Hermann, 1977

[Mar] J. Martinet: Singularities of Smooth Functions and Maps, Lectures notes Series 58, Cambridge University Press, 1982

[Mat] H. Matsumura: Commutative. ring theory, Cambridge University Press, 1986

[Mil] J. Milnor: Singular points of complex hypersurfaces, Princeton University Press, 1968

[Sin] Singular: G.-M. Greuel, G. Pfister, H. Schönemann. Available at URL=http://www.mathematik.uni-kl.de/ftp/pub/Math/Singular

[Tar] F. Tari: Recognition of K-Singularitoes of Functions, Experimental Mathematics, Vol 1 (1992), No.3