COMPUTATION OF THE MONODROMY OF AN ISOLATED HYPERSURFACE SINGULARITY

DIPLOMARBEIT BY MATHIAS SCHULZE

Abstract. This thesis deals with the implementation of an algorithm to compute the monodromy of an isolated hypersurface singularity in the computer algebra system Singular [GPS98]. We use an algorithm by E. Brieskorn [Bri70] from 1970 to compute a connection matrix of the meromorphic Gauss-Manin connection up to arbitraily high order and an algorithm by R. Gérard and A. H. M. Levelt [GL73] from 1973 to transform it to a simple pole.
In section 1 and 2, we develop the theory of *meromorphic differential equations* and *meromorphic connections*, in section 3, we introduce the *meromorphic Gauss-Manin connection* of an isolated hypersurface singularity, and, in section 4, we explain the implementation of the algorithm in Singular and compute some examples.

Contents

Introduction

In 1968, J. Milnor [Mil68] introduced the *local Picard-Lefschetz monodromy* of an isolated hypersurface singularity. Let

$$(\mathbb{C}^{n+1},0)=(X,x)\xrightarrow{f}(S,s)=(\mathbb{C},0)$$

be an isolated hypersurface singularity with X respectively S a ball of radius ϵ around x repectively a disc of radius δ around s and

$$\begin{aligned} S' &:= S\backslash\{s\},\\ X' &:= f^{-1}(S'). \end{aligned}$$

Milnor showed that for ϵ and δ sufficiently small

$$X'\xrightarrow{f}S'$$

is a locally trivial differentiable fibre bundle, and that the fibres

$$X_t := f^{-1}(t)$$

have the homotopy type of a bouquet of μ_f n-spheres where

$$\mu_f := \dim_{\mathbb{C}} \mathcal{O}_{X,x}/\langle \partial_{z_0} f,\ldots,\partial_{z_n} f\rangle$$

is the Milnor number of f. Hence,

$$\mathrm{H}^i(X_t,\mathbb{Z})\cong\begin{cases}\mathbb{Z}, & \text{if } i=0,\\ \mathbb{Z}^{\mu_f}, & \text{if } i=n,\\ 0, & \text{else.}\end{cases}$$

The fundamental group $\Pi_1(S',t)\cong\mathbb{Z}$ operates on the integer singular cohomology $\mathrm{H}(X_t,\mathbb{Z})$ and the automorphism

$$\mathrm{H}^n(X_t,\mathbb{Z})\xrightarrow{\mathrm{M}_f}\mathrm{H}^n(X_t,\mathbb{Z})$$

defined by a generator of $\Pi_1(S',t)$ is the *local Picard-Lefschetz monodromy* of f. It is a topological notion and determines the topology of the singularity to a certain extend.
E. Brieskorn [Bri70] gave an algebraic description of the *complex local Picard-Lefschetz monodromy*

$$\mathrm{H}^n(X_t,\mathbb{C})\xrightarrow{\mathrm{M}_f}\mathrm{H}^n(X_t,\mathbb{C})$$

using differential forms. Since f is locally trivial over S', the sheaf $\mathrm{R}^n f_*\mathbb{C}_{X'}$ associated to the presheaf

$$U\longmapsto \mathrm{H}^n(f^{-1}(U),\mathbb{C}_{X'})$$

is a locally constant sheaf on S' with $\mathrm{R}^n f_*\mathbb{C}_{X',t}=\mathrm{H}^n(X_t,\mathbb{C})$. The action of $\Pi_1(S',t)$ is given by shifting along local sections of $\mathrm{R}^n f_*\mathbb{C}_{X'}$.

The sheaf $\mathcal{O}_{S'} \otimes_{\mathbb{C}_{S'}} \mathrm{R}^n f_* \mathbb{C}_{X'}$ has a canonical integrable connection

$$\mathcal{O}_{S'} \otimes_{\mathbb{C}_{S'}} \mathrm{R}^n f_* \mathbb{C}_{X'} \xrightarrow{\nabla} \Omega^1_{S'} \otimes_{\mathbb{C}_{S'}} \mathrm{R}^n f_* \mathbb{C}_{X'}$$

defined by

$$\nabla(\alpha \otimes v) := \mathrm{d}\alpha \otimes v$$

and

$$\mathrm{R}^n f_* \mathbb{C}_{X'} = \ker \nabla \subset \mathcal{O}_{S'} \otimes_{\mathbb{C}_{S'}} \mathrm{R}^n f_* \mathbb{C}_{X'}.$$

Brieskorn used the isomorphisms

$$\mathcal{O}_{S'} \otimes_{\mathbb{C}_{S'}} \mathrm{R}^n f_* \mathbb{C}_{X'} \cong \mathrm{R}^n f_*(f^* \mathcal{O}_{S'}) \cong \mathbb{R}^n f_*(\Omega_{X'/S'})$$

to extend the sheaf $\mathcal{O}_{S'} \otimes_{\mathbb{C}_{S'}} \mathrm{R}^n f_* \mathbb{C}_{X'}$ on S' to the coherent sheaf $\mathbb{R}^n f_*(\Omega_{X/S})$ on S and ∇ to the regular singular integrable *Gauss-Manin connection*

$$\mathbb{R}^n f_*(\Omega_{X/S}) \xrightarrow{\nabla} \Omega^1_S \otimes_{\mathcal{O}_S} \mathbb{R}^n f_*(\Omega_{X/S}) \ .$$

He showed that $\mathbb{R}^n f_*(\Omega_{X/S})_s \cong \mathrm{H}^n(\Omega_{X/S,x})$, and that the covariant derivative of

$$\mathrm{H}^n(\Omega_{X/S,x}) \xrightarrow{\nabla_s} \Omega^1_{S,s} \otimes_{\mathcal{O}_{S,s}} \mathrm{H}^n(\Omega_{X/S,x})$$

is given by

$$\mathrm{H}^n(\Omega_{X/S,x}) \xrightarrow{\nabla_f} \mathrm{H}^n(\Omega_{X/S,x}) \ ,$$

$$[\omega] \longmapsto \left[\tfrac{\mathrm{d}\omega}{\mathrm{d}f}\right] \ .$$

We call ∇_f the the *meromorphic Gauss-Manin connection.* Brieskorn showed that its monodromy coincides with the complex local Picard-Lefschetz monodromy and gave an algorithm to compute it. We use this algorithm to compute a connection matrix of the meromorphic Gauss-Manin connection up to arbitrarily high order and an algorithm by R. Gérard and A. H. M. Levelt [GL73] to transform it to a simple pole.

There is already an implementation of a similar algorithm in the computer algebra system MAPLE V by P. F. M. Nacken [Nac90] from 1990, but it needs much more computation time and memory.

I should like to thank my thesis advisor Prof. Dr. G.-M. Greuel for many helpful discussions, Dr. H. Schönemann, Dr. O. Bachmann and K. Krüger for their advice and help concerning SINGULAR, T. Keilen for proof-reading this thesis, Dr. habil. B. Kreußler for his advice concerning literature, and Prof. Dr. A. H. M. Levelt for providing P. F. M. Nacken's master's thesis and his procedure for the current release of MAPLE V.

1. Meromorphic Differential Equations

In this section, we introduce *meromorphic differential equations* and define *equivalence*, *monodromy*, and *regularity* of a meromorphic differential equation. We show how these notions are related, develop different characterizations of regularity, and show how to compute the monodromy of a meromorphic differential equation with a *simple pole*.

We start with some preparations concerning the *exponential of matrices*, *multivalued functions*, and *linear differential equations*.
By $|\cdot|_k$, we denote the norm on $\mathbb{C}^m$ defined by

$$|x|_k := \sqrt[k]{\sum_{i=1}^{m} |x_i|^k}$$

for all $x \in \mathbb{C}^m$ and assume properties of these norms to be known.

Proposition 1.1 (The Exponential Map e)**.**

$$\mathrm{e}^T := \sum_{k=0}^{\infty} \frac{T^k}{k!}$$

defines a holomorphic map

$$\mathrm{Mat}(n,\mathbb{C}) \xrightarrow{\mathrm{e}} \mathrm{Mat}(n,\mathbb{C}).$$

Proof. Since

$$\left| \sum_{k=k_0}^{k_1} \frac{T^k}{k!} \right|_1 \le \sum_{k=k_0}^{k_1} \frac{|T|_1^k}{k!},$$

for all $k_0, k_1 \in \mathbb{N}$ with $k_0 \le k_1$, the locally uniform convergence of the complex exponential series implies that of $\sum_{k=0}^{\infty} \frac{T^k}{k!}$. □

By $\mathcal{O}_\mathbb{C}$ respectively $\mathcal{M}_\mathbb{C}$, we denote the sheaf of holomorphic respectively meromorphic functions on $\mathbb{C}$. We choose a complex coordinate t of $\mathbb{C}$ at 0 and identify

$$\begin{aligned} \mathcal{O}_{\mathbb{C},0} &= \mathbb{C}\{t\}, \\ \mathcal{M}_{\mathbb{C},0} &= \mathbb{C}\{t\}[t^{-1}] =: K. \end{aligned}$$

By $\sigma(C)$, we denote the *spectrum* of a matrix $C \in \mathrm{Mat}(n,\mathbb{C})$ and abbreviate by

$$\mathrm{Mat}_I(n,\mathbb{C}) := \{C \in \mathrm{Mat}(n,\mathbb{C}) | \mathrm{Re}(\sigma(C)) \subset I\}$$

the set of matrices with real part of the eigenvalues in the interval $I \subset \mathbb{R}$.
The following properties of e are well known.

Proposition 1.2 (Properties of e)**.**

1. $U\mathrm{e}^C U^{-1} = \mathrm{e}^{UCU^{-1}}$ for $C \in \mathrm{Mat}(n,\mathbb{C})$ and $U \in \mathrm{GL}_n(\mathbb{C})$.

2. $\mathrm{e}^{C+C'} = \mathrm{e}^C\mathrm{e}^{C'}$ for $C, C' \in \mathrm{Mat}(n,\mathbb{C})$ with $CC' = C'C$.
3. $\det \mathrm{e}^C = \mathrm{e}^{\mathrm{tr}\, C}$ for $C \in \mathrm{Mat}(n,\mathbb{C})$.
4. $\partial_t \mathrm{e}^A = \partial_t A \mathrm{e}^A$ for $A \in \mathrm{Mat}(n, \mathcal{O}_{\mathbb{C}}(U))$ with $U \subset \mathbb{C}$ open.
5. For $U \in \mathrm{GL}_n(\mathbb{C})$ and $I = (a,b] \subset \mathbb{R}$ or $I = [a,b) \subset \mathbb{R}$ with $b = a+1$, there is a unique $C \in \mathrm{Mat}_I(n,\mathbb{C})$ with $\mathrm{e}^{2\pi\mathrm{i}C} = U$.

We denote by

$$\mathbb{C}^* \overset{i}{\hookrightarrow} \mathbb{C}$$

with $i(t) := t$ the *inclusion* of $\mathbb{C}^*$ in $\mathbb{C}$. We choose a complex coordinate t' of $\mathbb{C}$ with

$$p(t') := \mathrm{e}^{2\pi\mathrm{i}t'} = t$$

and denote by

$$\mathbb{C} \overset{p}{\twoheadrightarrow} \mathbb{C}^*$$

the *universal covering* of $\mathbb{C}^*$. Another choice of t' leads to the same results.
Since p is locally biholomorphic and 1-periodic, we can consider $f \in \Gamma(U, i_*p_*\mathcal{O}_{\mathbb{C}})$ with $U \subset \mathbb{C}$ open as a *multivalued holomorphic function* $f(t) = f\left(\frac{\log t'}{2\pi\mathrm{i}}\right)$ on $i^{-1}(U)$, where $f(t) \in \Gamma(U, i_*\mathcal{O}_{\mathbb{C}^*})$ if and only if $f(t'+1) = f(t')$ for all $t' \in p^{-1}(i^{-1}(U))$.

Notation 1.3 (The Map t^C)**.**
We consider $\mathrm{e}^{2\pi\mathrm{i}Ct'} \in \mathrm{Mat}(n, \Gamma(\mathbb{C}, i_*p_*\mathcal{O}_{\mathbb{C}}))$ with $C \in \mathrm{Mat}(n,\mathbb{C})$ as a multivalued holomorphic map

$$t^C := \mathrm{e}^{C \log t'}.$$

The following corollary is an immediate consequence of proposition 1.2.

Corollary 1.4 (Properties of t^C)**.**
Let $C, C' \in \mathrm{Mat}(n,\mathbb{C})$.

1. If $U \in \mathrm{GL}(n,\mathbb{C})$, then $Ut^CU^{-1} = t^{UCU^{-1}}$.
2. If $CC' = C'C$, then $t^{C+C'} = t^Ct^{C'}$.
3. $\det t^C = t^{\mathrm{tr}\, C}$
4. $\partial_t C = \frac{C}{t}t^C$

We identify a *linear differential equation*

$$\partial_t + A = 0$$

with its defining matrix A.
The following theorem is a reformulation of the well known existence and uniqueness theorem for linear differential equations with holomorphic coefficients.

Theorem 1.5 (Existence and Uniqueness for Lin. Diff. Equ.).
If $A \in \mathrm{Mat}(n, \Gamma(U, \mathcal{O}_\mathbb{C}))$ with $U \subset \mathbb{C}$ open is a linear differential equation, then

$$\mathcal{S}_A := \ker(\partial_t + A) \subset \mathcal{O}_U^n$$

is a subsheaf of $\mathbb{C}$-vector spaces and evaluation induces local isomorphisms of $\mathcal{S}_A$ and the constant sheaf $\mathbb{C}^n$.

Proof. E. A. Coddington, N. Levinson [CL55], chapter 3, section 7 and theorem 2.1. □

Definition 1.6 (Solution Sheaf and Fundamental Matrices).
We call $\mathcal{S}_A$ the *solution sheaf* of A and a section $s \in \Gamma(V, \mathcal{S}_A)$ of $\mathcal{S}_A$ over an open subset $V \subset \mathbb{C}$ a *solution* of A over V. If U is simply connected, a matrix $F \in \mathrm{Mat}(n, \Gamma(U, \mathcal{O}_\mathbb{C}))$ the columns of which form a $\mathbb{C}$-basis of $\Gamma(U, \mathcal{S}_A) \cong \mathbb{C}^n$ is called a *fundamental matrix of* A.

Since the local isomorphisms in theorem 1.5 are induced by evaluation, there is a simple characterization of fundamental matrices.

Corollary 1.7 (Characterization of Fundamental Matrices).
If $A \in \mathrm{Mat}(n, \Gamma(U, \mathcal{O}_\mathbb{C}))$ with $U \subset \mathbb{C}$ open and simply connected is a linear differential equation, then $F \in \mathrm{Mat}(n, \Gamma(U, \mathcal{O}_\mathbb{C}))$ is a fundamental matrix of A if and only if $(\partial_t + A)F = 0$ and $F(t) \in \mathrm{GL}_n(\mathbb{C})$ for some and then for all $t \in U$.

Proof. E. A. Coddington, N. Levinson [CL55], chapter 3, section 7 and theorem 2.2. □

After these preparations, we study linear differential equations with meromorphic coefficients defined on an arbitrarily small neighbourhood of the origin.

Definition 1.8 (Meromorphic Differential Equations).
We call

$$\partial_t + A = 0$$

with $A \in \mathrm{Mat}(n, K)$ a *(n-dimensional) meromorphic differential equation.*

Let A be a meromorphic differential equation. Then there is an open neighbourhood $U \subset \mathbb{C}$ of 0 with $A \in \mathrm{Mat}(n, \Gamma(U, i_*\mathcal{O}_{\mathbb{C}^*}))$. Since $i^{-1}(U)$ is not simply connected, $\Gamma(U, i_*\mathcal{S}_A) = 0$ can occur and it makes no sense to consider $i_*\mathcal{S}_{A,0}$ as the *solution space* of A. Instead, we will define the *solution space* of A to be the space of *multivalued solutions* of A. If $s \in \Gamma(V, i_*\mathcal{S}_A)$ with $V \subset U$ open, then

$$\begin{aligned} 0 &= 2\pi\mathrm{i}pp^*(\partial_t s + As) \\ &= 2\pi\mathrm{i}pp^*(\partial_t s) + 2\pi\mathrm{i}pp^*Ap^*s \\ &= p^*(\partial_t s)\partial_t p + 2\pi\mathrm{i}p^*tp^*Ap^*s \\ &= \partial_t(p^*s) + p^*(2\pi\mathrm{i}tA)p^*s \end{aligned}$$

implies $p^*s \in \Gamma(V, i_*p_*\mathcal{S}_{p^*(2\pi\mathrm{i}tA)})$. Hence,

$$i_*\mathcal{S}_A \xhookrightarrow{p^*} i_*p_*\mathcal{S}_{p^*(2\pi\mathrm{i}tA)}.$$

Definition 1.9 (Solution Space and Multivalued Solutions)**.**
For a meromorphic differential equation A, we call

$$\mathrm{S}_A := i_*p_*\mathcal{S}_{p^*(2\pi\mathrm{i}tA),0}$$

with $p^*(2\pi\mathrm{i}tA) \in \mathrm{Mat}(n, i_*p_*\mathcal{O}_{\mathbb{C},0})$ the *solution space of* A and consider $s \in \mathrm{S}_A$ as a *multivalued solution* $s(t) = s\left(\frac{\log t'}{2\pi\mathrm{i}}\right)$ of A.

Hence, a multivalued solution of A is a solution of the lifted differential equation $p^*(2\pi\mathrm{i}tA)$ on the covering space.

Corollary 1.10 (Structure of the Solution Space)**.**
The solution space of an n-dimensional meromorphic differential equation A is an n-dimensional $\mathbb{C}$-vector space, i.e.

$$\mathrm{S}_A \cong \mathbb{C}^n.$$

Proof. Since the open discs $U \subset \mathbb{C}$ around 0 form a system of neighbourhoods of 0 with $p^{-1}(U)$ simply connected, the claim follows from theorem 1.5. □

Definition 1.11 ((Multivalued) Fundamental Matrices)**.**
For a meromorphic differential equation A, we consider a fundamental matrix $F \in \mathrm{Mat}(n, i_*p_*\mathcal{O}_{\mathbb{C},0})$ of $p^*(2\pi\mathrm{i}tA) \in \mathrm{Mat}(n, i_*p_*\mathcal{O}_{\mathbb{C},0})$ as a *(multivalued) fundamental matrix* $F(t) = F\left(\frac{\log t'}{2\pi\mathrm{i}}\right)$ of A.

By shifting a fundamental matrix on the covering space by one period, we get another fundamental matrix. This gives an automorphism of the solution space.

Proposition 1.12 (Monodromy of Merom. Diff. Equ.)**.**
If A is a meromorphic differential equation, then

$$\mathrm{M}_A(s) := s(t'-1)$$

defines an automorphism $\mathrm{M}_A \in \mathrm{Aut}_{\mathbb{C}}(\mathrm{S}_A)$ of the solution space of A which is called the *monodromy of* A.

Proof. Let F be a fundamental matrix of A. Since

$$\begin{aligned} 0 &= \partial_{t'}F(t'-1) + p^*(2\pi\mathrm{i}tA)(t'-1)F(t'-1) \\ &= \partial_{t'}F(t'-1) + p^*(2\pi\mathrm{i}tA)F(t'-1), \end{aligned}$$

both F and $F(t'-1)$ are fundamental matrices of A by corollary 1.7. □

Notation 1.13. For a fundamental matrix F of a meromorphic differential equation A, we denote by

$$\mathrm{M}_{A,F} := F^{-1}\mathrm{M}_A F = F^{-1}F(t'-1) \in \mathrm{GL}_n(\mathbb{C})$$

the *matrix* of M_A with respect to F and identify the automorphism M_A with the conjugation class of its matrix $\mathrm{M}_{A,F}$ with respect to F, i.e.

$$\mathrm{M}_A = [\mathrm{M}_{A,F}]$$

where $[\cdot]$ denotes the conjugation class.

A fundamental matrix can be factorized uniquely into a single valued factor and a multivalued factor corresponding to the monodromy.

Proposition 1.14 (Structure of Fundamental Matrices)**.**
A fundamental matrix of a meromorphic differential equation A is of the form St^C with unique $S \in \mathrm{Mat}(n, i_*\mathcal{O}_{\mathbb{C}^*,0})$ and $C \in \mathrm{Mat}_{(0,-1]}(n, \mathbb{C})$. Moreover, if St^C with $S \in \mathrm{Mat}(n, i_*\mathcal{O}_{\mathbb{C}^*,0})$ and $C \in \mathrm{Mat}(n, \mathbb{C})$ is a fundamental matrix of A, then

$$\mathrm{M}_{A,St^C} = \mathrm{e}^{-2\pi\mathrm{i}C}.$$

Proof. Let F be a fundamental matrix of A. Since $\mathrm{M}_{A,F} \in \mathrm{GL}_n(\mathbb{C})$, there is a unique $C \in \mathrm{Mat}_{(0,-1]}(n, \mathbb{C})$ with $\mathrm{e}^{-2\pi\mathrm{i}C} = \mathrm{M}_{A,F}$ by part 5 of proposition 1.2 and

$$F(t'-1)\mathrm{e}^{-2\pi\mathrm{i}C(t'-1)} = F\mathrm{M}_{A,F}\mathrm{e}^{2\pi\mathrm{i}C}\mathrm{e}^{-2\pi\mathrm{i}Ct'} = F\mathrm{e}^{-2\pi\mathrm{i}Ct'}$$

implies

$$S := Ft^{-C} \in \mathrm{Mat}(n, i_*\mathcal{O}_{\mathbb{C}^*,0}).$$

To prove uniqueness, let $S' \in \mathrm{Mat}(n, i_*\mathcal{O}_{\mathbb{C}^*,0})$ and $C' \in \mathrm{Mat}_{(0,-1]}(n, \mathbb{C})$ with $F = S't^{C'}$. Then

$$\mathrm{e}^{-2\pi\mathrm{i}C} = \mathrm{M}_{A,F} = \mathrm{e}^{-2\pi\mathrm{i}C'}$$

and, hence, $C = C'$ by uniqueness of C. This implies

$$S = St^Ct^{-C} = Ft^{-C} = Ft^{-C'} = S't^{C'}t^{-C'} = S'.$$

□

We will use the above representation of fundamental matrices without explicit reference.
We want to consider meromorphic differential equations only up to *meromorphic coordinate transformations*.

Definition 1.15 (Merom. Coord. Transf. and Equivalence)**.**
A matrix $U \in \mathrm{GL}_n(K)$ is called a *(n-dimensional) meromorphic coordinate transformation*.
Two meromorphic differential equations A and A' are called *equivalent* if there is a $U \in \mathrm{GL}_n(K)$ with

$$A' = A^U := U^{-1}(\partial_t U + AU)$$

and we write $A \sim A'$ in this case.

Remark 1.16 (Merom. Coord. Transf. and Fund. Mat.).
If F is a fundamental matrix of the meromorphic differential equation A and U a meromorphic coordinate transformation, then

$$\begin{aligned} 0 &= U^{-1}(\partial_t F + AF) \\ &= U^{-1}(\partial_t (UU^{-1}F) + AF) \\ &= \partial_t(U^{-1}F) + U^{-1}(\partial_t U + AU)U^{-1}F \\ &= \partial_t(U^{-1}F) + A^U U^{-1}F \end{aligned}$$

and, hence, $U^{-1}F$ is a fundamental matrix of A^U by corollary 1.7.

A meromorphic coordinate transformation may change the single valued factor of a fundamental matrix but preserves the multivalued factor and hence the monodromy.

Corollary 1.17 (Equivalence and Monodromy).
Equivalent meromorphic differential equations have the same monodromy, i.e.

$$A \sim A' \Rightarrow \mathrm{M}_A = \mathrm{M}_{A'}.$$

Proof. Let $F = St^C$ be a fundamental matrix of A and $U \in \mathrm{GL}_n(K)$. Then, $F' := U^{-1}St^C$ is a fundamental matrix of $A' := A^U$ by remark 1.16 and, hence,

$$\mathrm{M}_{A,F} = \mathrm{e}^{-2\pi \mathrm{i} C} = \mathrm{M}_{A',F'}$$

by proposition 1.14. □

In the following we consider *regular* meromorphic differential equations. *Regularity* is a condition on the growth of solutions near the origin.

Definition 1.18 (Regularity of Merom. Diff. Equ.).
A meromorphic differential equation A is called *regular* if $S \in \mathrm{GL}_n(K)$ for a fundamental matrix St^C of A.

Remark 1.19.

1. If St^C and $S't^{C'}$ are fundamental matrices of a meromorphic differential equation A, then, by corollary 1.10, there is a matrix $U \in \mathrm{GL}_n(\mathbb{C})$ with
$$S't^{C'} = St^C U = SUU^{-1}t^C U = SUt^{U^{-1}CU}$$
and, hence, $S' = SU$ and $C' = U^{-1}CU$ by uniqueness in proposition 1.14.
Hence, if A is regular, $S \in \mathrm{GL}_n(K)$ for all fundamental matrices St^C of A.
2. By remark 1.16, regularity is a property of the equivalence class.
3. If $F = St^C$ is a fundamental matrix of the meromorphic differential equation A with $S \in \mathrm{Mat}(n, K)$, then
$$\det S = \det F \det t^{-C} = \det F t^{-\operatorname{tr} C} \in K^*$$

by corollary 1.4 and corollary 1.7 and, hence, $S \in \mathrm{GL}_n(K)$. Hence, A is regular if $S \in \mathrm{Mat}(n, K)$ for a fundamental matrix St^C of A.

Regularity is characterized by the existence of a special representative in the equivalence class from which the monodromy can be computed easily.

Proposition 1.20 (Characterization of Regularity)**.**
A meromorphic differential equation A is regular if and only if

$$A \sim \frac{C}{t}$$

for some $C \in \mathrm{Mat}_{(0,1]}(n, \mathbb{C})$. Moreover, if $A \sim \frac{C}{t}$ for some $C \in \mathrm{Mat}(n, \mathbb{C})$, then

$$\mathrm{M}_A = [\mathrm{e}^{2\pi \mathrm{i} C}].$$

Proof. If A is regular with fundamental matrix St^{-C},

$$\begin{aligned}
\frac{C}{t} &= S^{-1}\big(\partial_t\big(St^{-C}\big) + ASt^{-C}\big)t^C + \frac{C}{t} \\
&= S^{-1}\left(\partial_t St^{-C} - S\frac{C}{t}t^{-C} + ASt^{-C} + S\frac{C}{t}t^{-C}\right)t^C \\
&= S^{-1}\big(\partial_t S + AS\big) \\
&= A^S
\end{aligned}$$

and, hence,

$$\frac{C}{t} \sim A$$

If $A \sim \frac{C}{t}$ with $C \in \mathrm{Mat}(n, \mathbb{C})$, then, since

$$\begin{aligned}
\partial_t t^{-C} &= -\frac{C}{t}t^{-C}, \\
\det t^{-C} &= t^{-\operatorname{tr} C}
\end{aligned}$$

by corollary 1.4, t^{-C} is a fundamental matrix of $\frac{C}{t}$ by corollary 1.7 and, hence, $A \sim \frac{C}{t}$ is regular by part 2 of remark 1.19. Moreover,

$$\mathrm{M}_A = [\mathrm{M}_{\frac{C}{t}, t^{-C}}] = [\mathrm{e}^{2\pi \mathrm{i} C}]$$

by proposition 1.14 □

We have seen already that the monodromy only depends on the equivalence class. As a consequence of the above result, a regular equivalence class is even determined by its monodromy.

Corollary 1.21 (Regular Equivalence and Monodromy)**.**
Two regular meromorphic differential equations are equivalent if and only if they have the same monodromy, i.e.

$$A \sim A' \Leftrightarrow \mathrm{M}_A = \mathrm{M}_{A'}.$$

Proof. By corollary 1.17, $A \sim A'$ implies $\mathrm{M}_A = \mathrm{M}_{A'}$.
To show the other implication, let $\mathrm{M}_A = \mathrm{M}_{A'}$. By proposition 1.20, we can assume $A = \frac{C}{t}$ and $A' = \frac{C'}{t}$ with $C, C' \in \mathrm{Mat}_{(0,1]}(n, \mathbb{C})$. Since

$$[\mathrm{M}_{\frac{C'}{t}, t^{-C'}}] = \mathrm{M}_{\frac{C}{t}} = \mathrm{M}_{\frac{C'}{t}} = [\mathrm{M}_{\frac{C}{t}, t^{-C}}],$$

there is a $U \in \mathrm{GL}_n(\mathbb{C})$ with

$$\begin{aligned} \mathrm{e}^{2\pi \mathrm{i} C'} &= \mathrm{M}_{\frac{C'}{t}, t^{-C'}} \\ &= U^{-1} \mathrm{M}_{\frac{C}{t}, t^{-C}} U \\ &= U^{-1} \mathrm{e}^{2\pi \mathrm{i} C} U \\ &= \mathrm{e}^{2\pi \mathrm{i} U^{-1} C U} \end{aligned}$$

and, hence, by part 5 of proposition 1.2,

$$A' = \frac{C'}{t} = \frac{U^{-1}CU}{t} = U^{-1}\left(\partial_t U + \frac{C}{t} U\right) = \left(\frac{C}{t}\right)^U = A^U.$$

□

The following proposition gives a weaker condition for regularity proposition 1.20.

Notation 1.22 ($C \oplus C'$ and $C \ominus C'$)**.**
For $C, C' \in \mathrm{Mat}(n, \mathbb{C})$, we denote by $C \oplus C', C \ominus C' \in \mathrm{End}_{\mathbb{C}}(\mathrm{Mat}(n, \mathbb{C})) = \mathrm{Mat}(n^2, \mathbb{C})$ the endomorphisms defined by

$$\begin{aligned} C \oplus C'(T) &:= CT + TC', \\ C \ominus C'(T) &:= CT - TC'. \end{aligned}$$

Proposition 1.23 (Characterization of Regularity)**.**
A meromorphic differential equation $\frac{B}{t}$ with $B \in \mathrm{Mat}(n, \mathbb{C}\{t\})$ is regular.

Proof. Let $t = \rho \mathrm{e}^{i\phi}$ and St^C a fundamental matrix of $\frac{B}{t}$. Then, using notation 1.22,

$$\begin{aligned} 0 &= \left(\partial_t (St^C) + \frac{B}{t} St^C\right) t^{-C} \\ &= \left(\partial_t S t^C + S\frac{C}{t} t^C + \frac{B}{t} St^C\right) t^{-C} \\ &= \partial_t S + S\frac{C}{t} + \frac{B}{t} S \\ &= \partial_t S + \frac{B \oplus C}{t} S \end{aligned}$$

and there is a $\rho_0 > 0$ with

$$|\partial_\rho S|_2 = \left|\mathrm{e}^{i\phi} \partial_t S\right|_2 = |\partial_t S|_2 \leq \frac{|B \oplus C|_2}{\rho} |S|_2 \leq \frac{N}{\rho} |S|_2$$

for all $0 < \rho \leq \rho_0$ and some $N \in \mathbb{N}$. Therefore,

$$\begin{aligned}\partial_\rho\big(\rho^N|S|_2\big) &= \rho^N\left(\frac{N}{\rho}|S|_2 + \partial_\rho|S|_2\right)\\ &\geq \rho^N\left(\frac{N}{\rho}|S|_2 - |\partial_\rho|S|_2|\right)\\ &\geq \rho^N\left(\frac{N}{\rho}|S|_2 - |\partial_\rho S|_2\right)\\ &\geq 0\end{aligned}$$

and, hence,

$$|t^N S|_2 = \rho^N|S|_2 \leq \rho_0^N \max\left|S\big(\rho_0 \mathrm{e}^{2\pi\mathrm{i}[0,1]}\big)\right|_2$$

for all $0 < \rho \leq \rho_0$. This implies $S \in \mathrm{Mat}(n, K)$ and hence regularity of $\frac{B}{t}$ by part 3 of remark 1.19. □

Definition 1.24 (Pole order of Merom. Diff. Equ.).
For a regular meromorphic differential equation A, we call

$$\mathrm{pol}\, A := \min\{k \in \mathbb{Z} | t^k A \in \mathrm{Mat}(n, \mathbb{C}\{t\})\}$$

the *pole order of* A and say A has a *simple pole* if $\mathrm{pol}\, A = 1$.

From proposition 1.20 and 1.23 we get the following equivalent characterizations of regularity.

Corollary 1.25 (Characterizations of Regularity).
For a meromorphic differential equation A, the following conditions are equivalent.

1. A is regular.
2. $A \sim \frac{C}{t}$ for some $C \in \mathrm{Mat}_{(0,1]}(n, \mathbb{C})$.
3. $A \sim \frac{B}{t}$ for some $B \in \mathrm{Mat}(n, \mathbb{C}\{t\})$.
4. $\mathrm{pol}(A^U) \leq 1$ for some $U \in \mathrm{GL}_n(K)$.

Definition 1.26 (Formal Solutions of Merom. Diff. Equ.).
For a meromorphic differential equation A, a formal power series vector $u \in \mathbb{C}[\![t]\!][t^{-1}]^n$ is called a *formal solution* of A if

$$(\partial_t + A)u = 0.$$

B. Malgrange [Mal74] gives another characterization of regularity, namely, that every formal solution is convergent. We show only one implication.

Proposition 1.27 (Formal Solutions of regular Merom. Diff. Equ.).
If A is a regular meromorphic differential equation and $u \in \mathbb{C}[\![t]\!][t^{-1}]^n$ a formal solution of A, then $u \in K^n$ and, in particular, $u \in \mathrm{S}_A$.

Proof. By proposition 1.20, we can assume $A = \frac{C}{t}$ after a meromorphic coordinate transformation. Since $\mathrm{GL}_n(\mathbb{C}) \subset \mathrm{Mat}(n, \mathbb{C})$ is an open subset and

$$\lim_{k\to\infty}\left(E - \frac{C}{k}\right) = E \in \mathrm{GL}_n(\mathbb{C}),$$

there is a $k_1 \in \mathbb{N}$ with

$$kE - C = k\left(E - \frac{C}{k}\right) \in \mathrm{GL}_n(\mathbb{C})$$

for all $k > k_1$. If $u = \sum_{k=k_0}^{\infty} u_k t^k$, then

$$\begin{aligned} 0 &= t\partial_t u + Cu \\ &= \sum_{k=k_0}^{\infty} k u_k t^k + C \sum_{k=k_0}^{\infty} u_k t^k \\ &= \sum_{k=k_0}^{\infty} (kE - C) u_k t^k \end{aligned}$$

and, hence,

$$u = \sum_{k=k_0}^{k_1} u_k t^k \in K^n.$$

□

Notation 1.28 (χ and δ)**.** By χ_C, we denote the *characteristic polynomial* of a matrix $C \in \mathrm{Mat}(n, \mathbb{C})$ and, by

$$\delta_C := \max((\sigma(C) - \sigma(C)) \cap \mathbb{Z}),$$

the *maximum integer difference of eigenvalues* of C.

We will use the following notation without explicit reference.

Notation 1.29 (Jets of Matrices)**.**
For a matrix $B \in \mathrm{Mat}(n, K)$, we denote

$$B =: \sum_{i=-\infty}^{\infty} B_i t^i$$

and, by

$$\mathrm{jet}_k B := \sum_{i=-\infty}^{k} B_i t^i \in \mathrm{Mat}(n, K),$$

the *k-jet* of B.

In the following, we prove that, for $B \in \mathrm{Mat}(n, \mathbb{C}\{t\})$,

$$\chi_{\mathrm{M}_{\frac{B}{t}}} = \chi_{\mathrm{e}^{2\pi\mathrm{i}B_0}}$$

and even

$$\mathrm{M}_{\frac{B}{t}} = [\mathrm{e}^{2\pi\mathrm{i}B_0}]$$

if $\delta_{B_0} = 0$. Hence, for *almost all* $B \in \mathrm{Mat}(n, \mathbb{C}\{t\})$, the monodromy $\mathrm{M}_{\frac{B}{t}}$ depends only on B_0.
First, we consider the case $\delta_{B_0} = 0$. The following lemma is easy to prove.

Lemma 1.30 (Spectrum of $C \oplus C'$ and $C \ominus C'$)**.**
For $C, C' \in \mathrm{Mat}(n, \mathbb{C})$,

$$\sigma(C \oplus C') = \sigma(C) + \sigma(C'),$$
$$\sigma(C \ominus C') = \sigma(C) - \sigma(C').$$

Proposition 1.31. If $\delta_{B_0} = 0$ for $B \in \mathrm{Mat}(n, \mathbb{C}\{t\})$, then $\frac{B}{t} \sim \frac{B_0}{t}$ and, in particular,

$$\mathrm{M}_{\frac{B}{t}} = [\mathrm{e}^{2\pi\mathrm{i}B_0}].$$

Proof. By lemma 1.30, $\delta_{B_0} = 0$ implies

$$\sigma((kE + B_0) \ominus B_0) = k + \sigma(B_0) - \sigma(B_0) \subset \mathbb{C}^*$$

for all $k \geq 1$ and, hence, $(kE + B_0) \ominus B_0 \in \mathrm{GL}_{n^2}(\mathbb{C})$. Hence, there is a unique $U \in \mathrm{Mat}(n, \mathbb{C}[\![t]\!])$ with $U_0 = E$ and

$$\begin{aligned}(kE + B_0)U_k - U_k B_0 &= ((kE + B_0) \ominus B_0)U_k \\ &= -\sum_{l=1}^{k} B_l U_{k-l}\end{aligned}$$

for all $k \geq 1$ or, equivalently,

$$\begin{aligned}0 &= \sum_{k=0}^{\infty}(k+1)U_{k+1}t^k + \sum_{k=0}^{\infty}\sum_{l=0}^{k+1}\frac{B_l}{t}U_{k+1-l}t^{k+1} - \sum_{k=0}^{\infty}U_{k+1}t^{k+1}\frac{B_0}{t} \\ &= \partial_t U + \frac{B}{t}U - U\frac{B_0}{t} \\ &= \partial_t U + \frac{B \ominus B_0}{t}U.\end{aligned}$$

Hence,

$$\frac{B_0}{t} = \left(\frac{B}{t}\right)^U$$

with $U \in \mathrm{Mat}(n, K)$ by proposition 1.27 and, in particular,

$$\mathrm{M}_{\frac{B}{t}} = [\mathrm{e}^{2\pi\mathrm{i}B_0}]$$

by proposition 1.20. □

In the case $\delta_B \geq 1$, there is a sequence of δ_B meromorphic coordinate transformations making $\delta_B = 0$ without increasing the pole order or changing the characteristic polynomial $\chi_{\mathrm{e}^{2\pi\mathrm{i}B_0}}$.

Proposition 1.32. If $\delta_{B_0} \geq 1$ for $B \in \mathrm{Mat}(n, \mathbb{C}\{t\})$, then there is a $U \in \mathrm{GL}_n(K)$ such that

$$\frac{B'}{t} := \left(\frac{B}{t}\right)^U$$

fulfills the following conditions.

1. $\mathrm{pol}\,\frac{B'}{t} \leq 1$

2. $\delta_{B'_0} = \delta_{B_0} - 1$
3. $\chi_{\mathrm{e}^{2\pi\mathrm{i}B'_0}} = \chi_{\mathrm{e}^{2\pi\mathrm{i}B_0}}$
4. $\frac{\mathrm{jet}_k B'}{t} = \mathrm{jet}_{k-1}\left(\left(\frac{\mathrm{jet}_{k+1} B}{t}\right)^U\right)$
5. $\mathrm{e}^{2\pi\mathrm{i}B'} = \mathrm{e}^{2\pi\mathrm{i}B}$ if $B = \begin{pmatrix} b_1 & & 0 \\ & \ddots & \\ 0 & & b_n \end{pmatrix} \in \mathrm{Mat}(n, \mathbb{C})$.

Proof. Let $\sigma(B_0)/\mathbb{Z} = \{[\lambda_1], \ldots, [\lambda_m]\}$ with $\mathrm{Re}\,\lambda_k = \min \mathrm{Re}[\lambda_k]$. After a linear coordinate change, we can assume

$$B = \begin{pmatrix} B^{1,1} & B^{1,2} \\ B^{2,1} & B^{2,2} \end{pmatrix}$$

with $B_0^{1,2} = B_0^{2,1} = 0$,

$$\sigma(B_0^{1,1}) = \{\lambda_1, \ldots, \lambda_m\},$$
$$\sigma(B_0^{2,2}) = \bigcup_{k=1}^{m} [\lambda_k] \setminus \{\lambda_k\}.$$

Let

$$U := \begin{pmatrix} tE & 0 \\ 0 & E \end{pmatrix} \in \mathrm{Mat}(n, K).$$

Then,

$$\begin{aligned} B' &= t\left(\frac{B}{t}\right)^U \\ &= tU^{-1}\left(\partial_t U + \frac{B}{t}U\right) \\ &= \begin{pmatrix} E & 0 \\ 0 & tE \end{pmatrix}\left(\begin{pmatrix} E & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} B^{1,1} & B^{1,2} \\ B^{2,1} & B^{2,2} \end{pmatrix}\begin{pmatrix} E & 0 \\ 0 & \frac{E}{t} \end{pmatrix}\right) \\ &= \begin{pmatrix} B^{1,1} + E & \frac{B^{1,2}}{t} \\ tB^{2,1} & B^{2,2} \end{pmatrix} \in \mathrm{Mat}(n, \mathbb{C}\{t\}) \end{aligned}$$

implies 1, 4 and 5 and, in particular,

$$B'_0 = \begin{pmatrix} B_0^{1,1} + E & B_1^{1,2} \\ 0 & B_0^{2,2} \end{pmatrix}$$

implies 2 and

$$\begin{aligned} \mathrm{e}^{2\pi\mathrm{i}B'_0} &= \mathrm{e}^{2\pi\mathrm{i}\begin{pmatrix} B_0^{1,1} & B_1^{1,2} \\ 0 & B_0^{2,2} \end{pmatrix}} \mathrm{e}^{2\pi\mathrm{i}\begin{pmatrix} E & 0 \\ 0 & 0 \end{pmatrix}} \\ &= \begin{pmatrix} \mathrm{e}^{2\pi\mathrm{i}B_0^{1,1}} & * \\ 0 & \mathrm{e}^{2\pi\mathrm{i}B_0^{2,2}} \end{pmatrix} \end{aligned}$$

implies 3. □

From proposition 1.31 and 1.32 we get the following corollaries.

Corollary 1.33. $\chi_{\mathrm{M}_{\frac{B}{t}}} = \chi_{\mathrm{e}^{2\pi\mathrm{i}B_0}}$ for $B \in \mathrm{Mat}(n, \mathbb{C}\{t\})$.

Corollary 1.34. $\mathrm{M}_{\frac{C}{t}} = [\mathrm{e}^{2\pi\mathrm{i}C}]$ for $C = \begin{pmatrix} c_1 & & 0 \\ & \ddots & \\ 0 & & c_n \end{pmatrix} \in \mathrm{Mat}(n, \mathbb{C})$.

The following example shows that the condition $\delta_{B_0} = 0$ in proposition 1.31 is necessary for $\mathrm{M}_{\frac{B}{t}} = [\mathrm{e}^{2\pi\mathrm{i}B_0}]$.

Example 1.35. Let us consider $B := \begin{pmatrix} 0 & t \\ 0 & 1 \end{pmatrix} \in \mathrm{Mat}(n, \mathbb{C}\{t\})$ with $\delta_{B_0} = 1$. Let $U := \begin{pmatrix} t & 0 \\ 0 & 1 \end{pmatrix}$ and $B' := \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$, then

$$\left(\frac{B}{t}\right)^U = \frac{B'}{t}$$

and, hence, $\frac{B}{t} \sim \frac{B'}{t}$. Clearly, $\chi_{\mathrm{e}^{2\pi\mathrm{i}B_0}} = \chi_{\mathrm{e}^{2\pi\mathrm{i}B'_0}}$ and, hence,

$$\chi_{\mathrm{M}_{\frac{B}{t}}} = \chi_{\mathrm{e}^{2\pi\mathrm{i}B_0}} = \chi_{\mathrm{e}^{2\pi\mathrm{i}B'_0}} = \chi_{\mathrm{M}_{\frac{B'}{t}}}$$

by corollary 1.33. This also follows from corollary 1.17. Since $\delta_{B'_0} = 0$,

$$\mathrm{M}_{\frac{B}{t}} = \mathrm{M}_{\frac{B'}{t}} = [\mathrm{e}^{2\pi\mathrm{i}B'_0}]$$

by corollary 1.17 and proposition 1.31. But, clearly, $[\mathrm{e}^{2\pi\mathrm{i}B'_0}] \neq [\mathrm{e}^{2\pi\mathrm{i}B_0}]$ and, hence,

$$\mathrm{M}_{\frac{B}{t}} \neq [\mathrm{e}^{2\pi\mathrm{i}B_0}].$$

2. Meromorphic Connections

In this section, we introduce a coordinate free notion of meromorphic differential equations, namely, *meromorphic connections*. A meromorphic connection corresponds to a meromorphic differential equation by choosing a basis. We define corresponding notions of *monodromy* and *regularity* for meromorphic connections and show how to transform a regular meromorphic differential equation to a simple pole.

For an n-dimensional K-vector space V, we identify $e \in \mathrm{Iso}_K(K^n, V)$ with its canonical image $(e_1, \ldots, e_n) \in V^n$.

Definition 2.1 (Meromorphic Connections)**.**
Let V be an n-dimensional K-vector space. We call a $\mathbb{C}$-linear map

$$V \xrightarrow{\nabla} V$$

a *(n-dimensional) meromorphic connection* on V if it fulfills the *Leibniz rule*, i.e.

$$\nabla(\alpha v) = \partial_t \alpha v + \alpha \nabla v$$

for all $\alpha \in K$ and $v \in V$. We call the pair $\nabla = (V, \nabla)$ a *(n-dimensional) meromorphic connection*. For a basis $e \in \mathrm{Iso}_K(K^n, V)$ of V, we denote by $\nabla_e \in \mathrm{Mat}(n, K)$ with

$$(\nabla_e)_{i,j} := (e^{-1} \nabla e_i)_j$$

the *(connection) matrix* of ∇ with respect to e. A *morphism of meromorphic connections*

$$(V, \nabla) \xrightarrow{\phi} (V', \nabla')$$

is defined by $\phi \in \mathrm{Hom}_K(V, V')$ such that

$$\begin{array}{ccc} V & \xrightarrow{\phi} & V' \\ \Big\downarrow \nabla & \circlearrowleft & \Big\downarrow \nabla' \\ V & \xrightarrow{\phi} & V' \end{array}$$

is a commutative diagram. For two bases $e \in \mathrm{Iso}_K(K^n, V)$ and $e' \in \mathrm{Iso}_K(K^n, V')$, we denote by $\phi_{e,e'} \in \mathrm{Mat}(n, K)$ the *matrix of ϕ with respect to e and e'*.

Let (V, ∇) be a meromorphic connection and $e \in \mathrm{Iso}_K(K^n, V)$. Then,

$$\begin{aligned} e^{-1} \nabla e v &= e^{-1} \nabla \left(\sum_{i=1}^{n} v_i e_i \right) \\ &= e^{-1} \left(\sum_{i=1}^{n} \partial_t v_i e_i + v_i \nabla e_i \right) \\ &= (\partial_t + \nabla_e) v \end{aligned}$$

for all $v \in K^n$ and, hence,

$$e^{-1}\nabla e = \partial_t + \nabla_e$$

or, equivalently,

$$\big(K^n, \partial_t + \nabla_e\big) \xrightarrow[\cong]{e} (V, \nabla)$$

is an isomorphism of meromorphic connections. Hence, with respect to e, the condition $\nabla = 0$ corresponds to the meromorphic differential equation

$$\partial_t + \nabla_e = 0.$$

Let

$$(V, \nabla) \xrightarrow[\cong]{\phi} (V', \nabla')$$

be an isomorphism of meromorphic connections and $e' \in \mathrm{Iso}_K(K^n, V')$. Then,

$$\begin{aligned}
\nabla_e v &= e^{-1}\nabla e v - \partial_t v \\
&= e^{-1}\phi^{-1}\phi\nabla\phi^{-1}\phi e v - \partial_t v \\
&= e^{-1}\phi^{-1}e'e'^{-1}\nabla' e' e'^{-1}\phi e v - \partial_t v \\
&= {\phi_{e,e'}}^{-1}e'^{-1}\nabla' e' \phi_{e,e'} v - \partial_t v \\
&= {\phi_{e,e'}}^{-1}(\phi_{e,e'}\partial_t v + \partial_t\phi_{e,e'} v + \nabla'_{e'}\phi_{e,e'} v) - \partial_t v \\
&= {\phi_{e,e'}}^{-1}(\partial_t\phi_{e,e'} + \nabla'_{e'}\phi_{e,e'}) v
\end{aligned}$$

for all $v \in K^n$ and, hence, using the notation from definition 1.15,

$$\nabla_e = {\phi_{e,e'}}^{-1}(\partial_t\phi_{e,e'} + \nabla'_{e'}\phi_{e,e'}) = (\nabla'_{e'})^{\phi_{e,e'}}.$$

This proves the following proposition.

Proposition 2.2 (Corresp. of Merom. Conn. and Diff. Equ.)**.**
The map

$$\nabla \longmapsto \nabla_e$$

with $\nabla = (V, \nabla)$ and $e \in \mathrm{Iso}_K(K^n, V)$ induces a bijection of isomorphism classes of n-dimensional meromorphic connections and equivalence classes of n-dimensional meromorphic differential equations.

The bijection in proposition 2.2 induces a notion of *monodromy* for meromorphic connections.

Definition 2.3 (Monodromy of Merom. Conn.)**.**
For a meromorphic connection $\nabla = (V, \nabla)$ and $e \in \mathrm{Iso}_K(K^n, V)$,

$$\mathrm{M}_\nabla := \mathrm{M}_{\nabla_e}$$

is called the *monodromy* of ∇.

For the notion of *regularity* induced by the bijection in proposition 2.2 for meromorphic connections we give an equivalent coordinate free definition.

Definition 2.4 (Lattices)**.** Let V be a finite dimensional K-vector space. A $\mathbb{C}\{t\}$-submodule $L \subset V$ with $\operatorname{rk}_{\mathbb{C}\{t\}} L = \dim_K V$ is called a *lattice in* V. We denote by $\operatorname{Lat}(V)$ the set of lattices in V.

The following remark is easy to prove.

Remark 2.5.

1. A lattice is a free $\mathbb{C}\{t\}$-module.
2. For an n-dimensional K-vector space V, $e \in \operatorname{Iso}_K(K^n, V)$ if and only if $e \in \operatorname{Iso}_{\mathbb{C}\{t\}}(\mathbb{C}\{t\}^n, L)$ for some $L \in \operatorname{Lat}(V)$. In particular, for $L, L' \in \operatorname{Lat}(V)$, $t^k L \subset L'$ for some $k \in \mathbb{Z}$.

Definition 2.6 (Pole order of Merom. Conn.)**.**
Let $\nabla = (V, \nabla)$ be a meromorphic connection. For $L \in \operatorname{Lat}(V)$, we call

$$\operatorname{pol}_L \nabla := \min\{k \in \mathbb{Z} | t^k \nabla L \subset L\}$$

the *pole order* of ∇ on L and ∇ is said to have a *simple pole* on L if $\operatorname{pol}_L \nabla = 1$. In this case, the endomorphism $\operatorname{res}_L \nabla \in \operatorname{End}_{\mathbb{C}}(L/tL)$ induced by $t\nabla$ is called the *residue* of ∇ on L. We call

$$\operatorname{pol} \nabla := \min\{\operatorname{pol}_L \nabla | L \in \operatorname{Lat}(V)\}$$

the *pole order* of ∇ and ∇ is called *regular* if $\operatorname{pol} \nabla \leq 1$.

The minima in this definition exist by the following remark.

Remark 2.7. Let $\nabla = (V, \nabla)$ be a meromorphic connection and $L \in \operatorname{Lat}(V)$. Then, using definition 1.24,

$$\operatorname{pol}_L \nabla = \max\{0, \operatorname{pol} \nabla_e\}$$

for all $e \in \operatorname{Iso}_{\mathbb{C}\{t\}}(\mathbb{C}\{t\}^n, L)$ and

$$\operatorname{pol} \nabla = \max\{0, \min\{\operatorname{pol} \nabla_e | e \in \operatorname{Iso}_K(K^n, V)\}\}.$$

Proof. For all $e \in \operatorname{Iso}_{\mathbb{C}\{t\}}(\mathbb{C}\{t\}^n, L)$,

$$\begin{aligned} & t^k \nabla L \subset L \\ \Leftrightarrow\quad & t^k e^{-1} \nabla e e^{-1} L \subset e^{-1} L \\ \Leftrightarrow\quad & t^k (\partial_t + \nabla_e) \mathbb{C}\{t\}^n \subset \mathbb{C}\{t\}^n \\ \Leftrightarrow\quad & 0 \leq k \geq \operatorname{pol} \nabla_e \end{aligned}$$

and, hence,

$$\operatorname{pol}_L \nabla = \max\{0, \operatorname{pol} \nabla_e\}.$$

The second satement follows from the definition. $\square$

By propostion 1.25 and this remark, the notions of regularity for meromorphic connections and meromorphic differential equations coincide with respect to the bijection in proposition 2.2.

Proposition 2.8 (Regularity of Merom. Conn. and Diff. Equ.)**.**
If $\nabla = (V, \nabla)$ is a meromorphic connection and $e \in \operatorname{Iso}_K(K^n, V)$, then ∇ is regular if and only if ∇_e is regular.

The following proposition shows how to compute a lattice on which a regular meromorphic connection has at most a simple pole.

Proposition 2.9 (Existence of $t\nabla$-Stable Lattices)**.**
If $\nabla = (V, \nabla)$ is a regular meromorphic connection and $L \in \mathrm{Lat}(V)$, then

$$L_\infty := L + t\nabla L + (t\nabla)^2 L + \cdots$$

is a lattice, i.e. $L_\infty \in \mathrm{Lat}(V)$. In particular,

$$\mathrm{pol}_{L_\infty} \nabla \leq 1.$$

Proof. Since ∇ is regular, $t\nabla R \subset R$ for some $R \in \mathrm{Lat}(V)$. By part 2 of remark 2.5, $t^k L \subset R$ for some $k \in \mathbb{Z}$. Let

$$\begin{aligned} L_0 &: = L, \\ L_i &: = L_{i-1} + t\nabla L_{i-1} \text{ for } i \in \mathbb{N}. \end{aligned}$$

Assuming $L_{i-1} \in \mathrm{Lat}(V)$, the Leibniz rule implies, that L_i is a $\mathbb{C}\{t\}$-module. Hence, since $L_{i-1} \subset L_i$, $L_i \in \mathrm{Lat}(V)$. By induction,

$$L_0 \subset L_1 \subset L_2 \subset \cdots \subset V$$

is an ascending chain of lattices in V. Since by definition

$$L_i = L + t\nabla L + (t\nabla)^2 L + \cdots + (t\nabla)^i L,$$

the claim is equivalent to this chain or equivalently the chain

$$t^k L_0 \subset t^k L_1 \subset t^k L_2 \subset \cdots \subset V,$$

being stationary. Since the Leibniz rule implies

$$\begin{aligned} t^k L_i &= t^k L_{i-1} + t^k t\nabla L_{i-1} \\ &= t^k L_{i-1} + t\nabla(t^k L_{i-1}) \end{aligned}$$

for all $i \in \mathbb{N}$, we have reduced the claim to the case $L_i \subset R$ for all $i \in \mathbb{N}$. But in this case,

$$L_0 \subset L_1 \subset L_2 \subset \cdots \subset R$$

is an ascending chain of $\mathbb{C}\{t\}$-submodules of R which is stationary since R is a finite module over the noetherian ring $\mathbb{C}\{t\}$ and hence itself noetherian. $\square$

R. Gérard and A. H. M. Levelt proved the following estimation.

Theorem 2.10 (Estimation for $t\nabla$-Stable Lattices)**.**
If (V, ∇) is an n-dimensional regular meromorphic connection and $L \in \mathrm{Lat}(V)$, then $L_\infty = L_{n-1}$.

Proof. R. Gérard, A. H. M. Levelt [GL73], théorème 4.2. $\square$

We reformulate proposition 2.9 and theorem 2.10 in terms of a basis and give estimations for the computation.

Notation 2.11 (Jets of Modules).
For a $\mathbb{C}\{t\}$-submodule M of K^n and $k \in \mathbb{Z}$, we denote by $\operatorname{jet}_k M$ the $\mathbb{C}\{t\}$-submodule of K^n generated by the subset $\operatorname{jet}_k M \subset K^n$.

Corollary 2.12 (Computation of Transf. to Simple Pole).
Let $\frac{M}{t^k}$ with $M \in \operatorname{Mat}(n, \mathbb{C}\{t\})$ and $k \in \mathbb{N}$ be a regular meromorphic differential equation. Then,

$$
\begin{aligned}
L_0 &:= t^{(k-1)(n-1)}\mathbb{C}\{t\}^n, \\
L_i &:= L_{i-1} + \operatorname{jet}_{(k-1)(n-1)}\left(\frac{\operatorname{jet}_{(k-1)i} M}{t^{k-1}} L_{i-1}\right) \text{ for } i \in \mathbb{N}
\end{aligned}
$$

defines an increasing sequence of lattices in K^n,

$$i_\infty := \min\{i | L_i = L_{i+1}\} \leq n-1,$$

$L_\infty := L_{i_\infty}$ is a lattice in $\mathbb{C}\{t\}^n \subset K^n$, and

$$\operatorname{pol}\left(\frac{M}{t^k}\right)^U \leq 1$$

for $U \in \operatorname{Iso}_{\mathbb{C}\{t\}}(\mathbb{C}\{t\}^n, L_\infty) \subset \operatorname{GL}_n(K)$.
Moreover, U can be chosen to fulfill

$$U = \big(\operatorname{jet}_{(k-1)(n-1)} - \operatorname{jet}_{(k-1)(n-i_\infty-1)-1}\big)U.$$

In the following, we show that it is enough to define a meromorphic connection on a lattice.

Definition 2.13 (Holomorphic Connections).
If L and L' are free $\mathbb{C}\{t\}$-modules of rank n, we call a $\mathbb{C}$-linear map

$$L \xrightarrow{\nabla'} L'$$

a *(n-dimensional) holomorphic connection* if it fulfills the *Leibniz rule*, i.e.

$$\nabla'(gv) = \partial_t g v + g\nabla' v$$

for all $g \in \mathbb{C}\{t\}$ and $v \in L$.

Proposition 2.14 (Associated Merom. Conn.).
Let V be a finite dimensional K-vector space, $L, L' \in \operatorname{Lat}(V)$, and

$$L \xrightarrow{\nabla'} L'$$

a holomorphic connection. Then, ∇' extends to a unique meromorphic connection (V, ∇) which we call the *associated meromorphic connection* to ∇'.

Proof. For each $v \in V$, we can choose a $k \in \mathbb{Z}$ with $t^k v \in L$. Since ∇' fulfills the Leibniz rule,

$$\nabla v := -\frac{k}{t} v + t^{-k}\nabla'(t^k v)$$

is independent of the choice of k and (V, ∇) is the unique meromorphic connection extending ∇'. $\square$

3. The Meromorphic Gauss-Manin Connection

In this section we introduce the *meromorphic Gauss-Manin connection* of an isolated hypersurface singularity.

Let

$$(\mathbb{C}^{n+1},0)=(X,x)\xrightarrow{f}(S,s)=(\mathbb{C},0)$$

be an *isolated hypersurface singularity.* We choose a complex coordinate system $z=z_0,\dots,z_n$ of $\mathbb{C}^{n+1}$ at 0 and identify $\mathcal{O}_{\mathbb{C}^{n+1},0}=\mathbb{C}\{z\}$. We abbreviate

$$\partial_z f:=\partial_{z_0}f,\dots,\partial_{z_n}f.$$

By $\mathfrak{m}_{S,s}$ respectively $\mathfrak{m}_{X,x}$, we denote the *maximal ideal* of $\mathcal{O}_{S,s}$ respectively $\mathcal{O}_{X,x}$ and, by $\langle\partial_z f\rangle_{\mathbb{C}\{z\}}$, the *Jacobian ideal* of f.
Since

$$\sqrt{\langle\partial_z f\rangle_{\mathbb{C}\{z\}}}=\mathrm{I}(\mathrm{V}(\langle\partial_z f\rangle_{\mathbb{C}\{z\}}))=\mathrm{I}(\{0\})=\mathfrak{m}_{X,x}=\langle z\rangle_{\mathbb{C}\{z\}}$$

by the *Hilbert-Rückert Nullstellensatz*,

$$\langle z\rangle^k_{\mathbb{C}\{z\}}\subset\langle\partial_z f\rangle_{\mathbb{C}\{z\}}$$

for some $k\in\mathbb{N}$. Hence, by the *determinacy theorem*, we can assume $f\in\mathbb{C}[z]$ after a holomorphic coordinate transformation.
Since $\langle z\rangle^k_{\mathbb{C}}\subset\langle\partial_z f\rangle_{\mathbb{C}\{z\}}$, the *Milnor algebra* $\mathbb{C}\{z\}/\langle\partial_z f\rangle$ of f is a finite dimensional $\mathbb{C}$-vector space. Its dimension

$$\mu_f:=\dim_{\mathbb{C}}\mathbb{C}\{z\}/\langle\partial_z f\rangle<\infty$$

is called the *Milnor number* of f.
Since $\langle z\rangle^k_{\mathbb{C}\{z\}}\subset\langle\partial_z f\rangle_{\mathbb{C}\{z\}}$ and $f\in\langle z\rangle_{\mathbb{C}\{z\}}$, we can define κ_f to be minimal with

$$f^{\kappa_f}\in\langle\partial_z f\rangle_{\mathbb{C}\{z\}}.$$

Proposition 3.1. There is a $\xi'=\frac{\xi}{u}\in\mathbb{C}[z]^n_{\langle z\rangle}$ with

$$f^{\kappa_f}=\sum_{i=0}^n\xi'_i\partial_{z_i}f.$$

In particular,

$$f^{\kappa_f}\mathrm{d}z=\mathrm{d}f\wedge\xi'$$

where we identify

$$\xi'=\sum_{i=0}^n(-1)^i\xi'_i\mathrm{d}z_{\hat{i}}.$$

Proof. Since $\mathbb{C}[\![z]\!]$ is a flat $\mathbb{C}[z]_{\langle z\rangle}$-module by D. Eisenbud [Eis96], theorem 7.2,

$$f^{\kappa_f} \in \ker\big(\,\mathbb{C}[\![z]\!] \longrightarrow\!\!\!\!\!\to \mathbb{C}[\![z]\!]/\langle\partial_z f\rangle\,\big) = \ker\big(\,\mathbb{C}[z]_{\langle z\rangle} \longrightarrow\!\!\!\!\!\to \mathbb{C}[z]_{\langle z\rangle}/\langle\partial_z f\rangle\,\big)\otimes_{\mathbb{C}[z]_{\langle z\rangle}} \mathbb{C}[\![z]\!]\,.$$

Hence, by D. Eisenbud [Eis96], theorem 7.1,

$$\ker\big(\,\mathbb{C}[\![z]\!] \longrightarrow\!\!\!\!\!\to \mathbb{C}[\![z]\!]/\langle\partial_z f\rangle\,\big)/\langle z\rangle^i = \ker\big(\,\mathbb{C}[z]_{\langle z\rangle} \longrightarrow\!\!\!\!\!\to \mathbb{C}[z]_{\langle z\rangle}/\langle\partial_z f\rangle\,\big)/\langle z\rangle^i$$

for all $i \in \mathbb{N}$. Since $f \in \mathbb{C}[z]$, this implies

$$f^{\kappa_f} \in \ker\big(\,\mathbb{C}[z]_{\langle z\rangle} \longrightarrow\!\!\!\!\!\to \mathbb{C}[z]_{\langle z\rangle}/\langle\partial_z f\rangle\,\big).$$

□

We denote by $(\Omega_{\mathbb{C}^{n+1}}, \mathrm{d})$ the complex of sheafs of *holomorphic differential forms* on $\mathbb{C}^{n+1}$ and by (Ω_f, d_f) with

$$\Omega_f := \Omega_{X,x}/\mathrm{d}f \wedge \Omega_{X,x}$$

and d_f induced by d the complex of *relative differential forms* on X at 0.

Definition 3.2 (Brieskorn Lattices)**.** The $\mathcal{O}_{S,s}$-modules

$$\begin{aligned} H_f &:= \mathrm{H}^n(\Omega_f, \mathrm{d}_f),\\ H'_f &:= \operatorname{coker} \mathrm{d}_f^{n-1} = \Omega^n_{X,x}/(\mathrm{d}f \wedge \Omega^{n-1}_{X,x} + \mathrm{d}\Omega^{n-1}_{X,x}),\\ H''_f &:= \Omega^{n+1}_{X,x}/\mathrm{d}f \wedge \mathrm{d}\Omega^{n-1}_{X,x} \end{aligned}$$

are called *Brieskorn lattices.*

The following proposition was conjectured by Brieskorn and proved by M. Sebastiani.

Proposition 3.3. H_f, H'_f, and H''_f are free $\mathcal{O}_{S,s}$-modules of rank μ_f.

Proof. M. Sebastiani [Seb70], Corollaire 1. □

Since $\mathrm{d}f \wedge \mathrm{d}f \wedge \Omega_{X,x} = 0$, f defines a complex $(\Omega_{X,x}, \mathrm{d}f \wedge \cdot)$ which is called the *de Rham complex* of f. It is exact except at the extreme right.

Proposition 3.4 (De Rham Lemma)**.**

$$0 \longrightarrow (\Omega_{X,x}, \mathrm{d}f \wedge \cdot) \longrightarrow \mathcal{O}_{X,x}/\langle\partial_z f\rangle \longrightarrow 0$$

is an exact sequence of $\mathcal{O}_{X,x}$-modules.

Proof. Since $\mathbb{C}\{z\}$ is a regular local ring of dimension $n+1$, z is a $\mathbb{C}\{z\}$-sequence. By D. Eisenbud [Eis96], corollary 17.8,

$$z^k := z_0^k, \dots, z_n^k \in \langle \partial_z f \rangle_{\mathbb{C}\{z\}}$$

is a $\mathbb{C}\{z\}$-sequence and, hence, by D. Eisenbud [Eis96], corollary 17.7, $\partial_z f$ is a $\mathbb{C}\{z\}$-sequence. Denoting by

$$\mathrm{Kos}(\partial_z f) := \left(\bigwedge \mathbb{C}\{z\}^n, \partial_z f \wedge \cdot \right)$$

the *Koszul complex* defined by $\partial_z f$,

$$0 \longrightarrow \mathrm{Kos}(\partial_z f) \longrightarrow \mathbb{C}\{z\}/\langle \partial_z f \rangle \longrightarrow 0,$$

is an exact sequence of $\mathbb{C}\{z\}$-modules by D. Eisenbud [Eis96], corollary 17.5. Since

$$\Omega^1_{X,x} \cong \bigoplus_{i=0}^{n} \mathbb{C}\{z\} \mathrm{d}z_i \cong \mathbb{C}\{z\}^n$$

induces an isomorphism

$$(\Omega_{X,x}, \mathrm{d}f \wedge \cdot) \cong \mathrm{Kos}(\partial_z f),$$

this implies the claim. □

Notation 3.5. We abbreviate

$$\begin{aligned} \mathrm{d}z &:= \mathrm{d}z_0 \wedge \cdots \wedge \mathrm{d}z_n \in \Omega^{n+1}_{X,x}, \\ \mathrm{d}z_{\hat{i}} &:= \mathrm{d}z_0 \wedge \cdots \wedge \mathrm{d}z_{i-1} \wedge \mathrm{d}z_{i+1} \wedge \cdots \wedge \mathrm{d}z_n \in \Omega^n_{X,x}, \\ \mathrm{d}z_{\hat{i},\hat{j}} &:= \mathrm{d}z_0 \wedge \cdots \wedge \mathrm{d}z_{i-1} \wedge \mathrm{d}z_{i+1} \wedge \cdots \wedge \mathrm{d}z_{j-1} \wedge \mathrm{d}z_{j+1} \wedge \cdots \wedge \mathrm{d}z_n \in \Omega^{n-1}_{X,x}. \end{aligned}$$

Corollary 3.6. The maps

$$\Omega^n_{X,x} \xrightarrow{\mathrm{id}} \Omega^n_{X,x} \xrightarrow{\mathrm{d}f \wedge \cdot} \Omega^{n+1}_{X,x}$$

induce inclusions

$$H_f \overset{\mathrm{id}}{\hookrightarrow} H'_f \overset{\mathrm{d}f\wedge\cdot}{\hookrightarrow} H''_f$$

with $t^{\kappa_f} H''_f \subset \mathrm{d}f \wedge H'_f$ and $t^{\kappa_f} H'_f \subset H_f$. Moreover, the cokernels of both inclusions are isomorphic to $\mathbb{C}\{z\}/\langle \partial_z f \rangle$.

Proof. Clearly,

$$\Omega^n_{X,x} \xrightarrow{\mathrm{id}} \Omega^n_{X,x}$$

induces an inclusion

$$H_f \overset{\mathrm{id}}{\hookrightarrow} H'_f.$$

If $\omega \in \Omega^n_{X,x}$, then, by proposition 3.1,

$$\begin{aligned} \mathrm{d}(f^{\kappa_f}\omega) &= \kappa_f f^{\kappa_f-1}\mathrm{d}f \wedge \omega + f^{\kappa_f}\mathrm{d}\omega \\ &= \mathrm{d}f \wedge \left(\kappa_f f^{\kappa_f-1}\omega + \frac{\mathrm{d}\omega}{\mathrm{d}z}\xi'\right) \end{aligned}$$

and, hence, $t^{\kappa_f}H'_f \subset H_f$.
If $\omega \in \Omega^n_{X,x}$ with $\mathrm{d}f \wedge \omega = \mathrm{d}f \wedge \mathrm{d}\omega'$ for some $\omega' \in \Omega^{n-1}_{X,x}$, then, by the de Rham lemma, there is a $\omega'' \in \Omega^{n-1}_{X,x}$ with $\omega = \mathrm{d}f \wedge \omega'' + \mathrm{d}\omega'$ and, hence,

$$\Omega^n_{X,x} \xrightarrow{\mathrm{d}f\wedge\cdot} \Omega^{n+1}_{X,x}$$

induces an inclusion

$$H'_f \xhookrightarrow{\mathrm{d}f\wedge\cdot} H''_f.$$

If $\omega \in \Omega^{n+1}_{X,x}$, then

$$f^{\kappa_f}\omega = \mathrm{d}f \wedge \left(\frac{\omega}{\mathrm{d}z}\xi'\right)$$

by proposition 3.1 and, hence, $t^{\kappa_f}H''_f \subset \mathrm{d}f \wedge H'_f$.
Moreover,

$$\begin{aligned} H'_f/H_f &= (\Omega^n_f/\operatorname{im}\mathrm{d}^{n-1}_f)/(\ker \mathrm{d}^n_f/\operatorname{im}\mathrm{d}^{n-1}_f) \\ &\cong \Omega^n_f/\ker \mathrm{d}^n_f \\ &\cong \operatorname{im}\mathrm{d}^n_f \\ &= \Omega^{n+1}_f \\ &= \Omega^{n+1}_{X,x}/\mathrm{d}f \wedge \Omega^n_{X,x} \\ &\cong \mathcal{O}_{X,x}/\langle \partial_z f\rangle \end{aligned}$$

and $H''_f/\mathrm{d}f \wedge H'_f \cong \Omega^{n+1}_{X,x}/\mathrm{d}f \wedge \Omega^n_{X,x}$. □

Let

$$V := H''_f \otimes_{\mathbb{C}\{t\}} K.$$

Remark 3.7. By proposition 3.3 and corollary 3.6, V is a μ_f dimensional K-vector space and H_f, H'_f, and H''_f are lattices in V.

Proposition 3.8. Using definition 2.13,

$$\nabla'_f\big[\mathrm{d}f \wedge \omega\big] := \big[\mathrm{d}\omega\big]$$

defines a holomorphic connection

$$\mathrm{d}f \wedge H'_f \xrightarrow{\nabla'_f} H''_f.$$

Proof. If $\omega \in \Omega^n_{X,x}$ with $\mathrm{d}f \wedge \omega = \mathrm{d}f \wedge \mathrm{d}\omega'$ for some $\omega' \in \Omega^{n-1}_{X,x}$, then, by the de Rham lemma, $\omega = \mathrm{d}\omega' + \mathrm{d}f \wedge \omega''$ for some $\omega'' \in \Omega^{n-1}_{X,x}$ and, hence, $\mathrm{d}\omega = \mathrm{d}f \wedge \mathrm{d}\omega''$. Hence, ∇'_f is well defined.
Clearly, ∇'_f is $\mathbb{C}$-linear.

Since

$$\begin{aligned}\nabla'_f\big(g\big[\mathrm{d}f\wedge\omega\big]\big) &= \nabla'_f\big[\mathrm{d}f\wedge\big(f^*g\omega\big)\big]\\ &= \big[\mathrm{d}\big(f^*g\omega\big)\big]\\ &= \big[\mathrm{d}\big(f^*g\big)\wedge\omega+f^*g\mathrm{d}\omega\big]\\ &= \big[f^*(\partial_t g)\mathrm{d}f\wedge\omega+f^*g\mathrm{d}\omega\big]\\ &= \partial_t g\big[\mathrm{d}f\wedge\omega\big]+g\nabla'_f\big[\mathrm{d}f\wedge\omega\big]\end{aligned}$$

for $g\in\mathbb{C}\{t\}$ and $\omega\in\Omega^n_{X,x}$, ∇'_f fulfills the Leibniz rule. ☐

Definition 3.9 (The Meromorphic Gauss-Manin Connection)**.**
Using proposition 3.3, we call the associated meromorphic connection

$$\nabla_f=(V,\nabla_f)$$

of ∇'_f the *meromorphic Gauss-Manin connection of f* and denote its monodromy by

$$\mathrm{M}_f:=\mathrm{M}_{\nabla_f}$$

and the characteristic polynomial of its monodromy by

$$\Delta_f:=\chi_{\mathrm{M}_f}.$$

Proposition 3.10. ∇_f is regular.

Proof. E. Brieskorn [Bri70], Satz 2. ☐

Proposition 3.11. Δ_f is a product of cyclotomic polynomials.

Proof. E. Brieskorn [Bri70], Satz 4. ☐

Since $t^{\kappa_f}H''_f\subset\mathrm{d}f\wedge H'_f$ by corollary 3.6, $t^{\kappa_f}\nabla_f$ restricts to a $\mathbb{C}$-linear map

$$H''_f\xrightarrow{t^{\kappa_f}\nabla_f}H''_f$$

which we compute explicitly in terms of representative differential forms of H''_f.

Proposition 3.12 (Computing $t^{\kappa_f}\nabla_f$ on H''_f)**.**
The $\mathbb{C}$-linear map

$$H''_f\xrightarrow{t^{\kappa_f}\nabla_f}H''_f$$

is given by

$$t^{\kappa_f}\nabla_f[e\mathrm{d}z]=\left[\left(\sum_{i=0}^n\partial_{z_i}(e\xi'_i)-\kappa_f f^{\kappa_f-1}e\right)\mathrm{d}z\right]$$

with ξ' from propostion 3.1.

Proof. By propostion 3.1,

$$\begin{aligned} t^{\kappa_f}\nabla_f[e\mathrm{d}z] &= t^{\kappa_f}\nabla_f\big(t^{-\kappa_f}\big[ef^{\kappa_f}\mathrm{d}z\big]\big)\\ &= -\frac{\kappa_f}{t}\big[ef^{\kappa_f}\mathrm{d}z\big]+\nabla'_f[\mathrm{d}f\wedge(e\xi')]\\ &= -\kappa_f t^{\kappa_f-1}\big[e\mathrm{d}z\big]+\big[\mathrm{d}(e\xi')\big]\\ &= \big[\mathrm{d}(e\xi')-\kappa_f f^{\kappa_f-1}e\mathrm{d}z\big]\\ &= \left[\sum_{i=0}^{n}\mathrm{d}\big((-1)^i e\xi'_i\mathrm{d}z_{\hat{i}}\big)-\kappa_f f^{\kappa_f-1}e\mathrm{d}z\right]\\ &= \left[\left(\sum_{i=0}^{n}\partial_{z_i}(e\xi'_i)-\kappa_f f^{\kappa_f-1}e\right)\mathrm{d}z\right]. \end{aligned}$$

□

By $\langle\cdot,\cdot\rangle$, we denote the standard scalar product of $\mathbb{R}^{n+1}$.
If f is quasihomogeneous, then $\kappa_f=1$, ξ' is given by the *Euler relation*, and we can easily compute $t^{\kappa_f}\nabla_f$ on H''_f using proposition 3.12.

Corollary 3.13 (Computing the Monodromy for Quasihomog. f)**.** Let

$$f=\sum_{\langle\alpha,w\rangle=d} f_\alpha z^\alpha\in\mathbb{C}[z]$$

with $w\in\mathbb{N}^{n+1}$ and $d\in\mathbb{N}$ be quasihomogeneous and let $e=(z^{\alpha^i})_{i=1,\dots,\mu_f}$ represent a monomial $\mathbb{C}$-basis of $\mathbb{C}\{z\}/\langle\partial_z f\rangle\xrightarrow[\cong]{\mathrm{d}z}\Omega^{n+1}_{X,x}/\mathrm{d}f\wedge\Omega^n_{X,x}$. Then, $\kappa_f=1$, $e\mathrm{d}z$ represents an $\mathcal{O}_{S,s}$-basis of H''_f, and

$$t\nabla_{f,te\mathrm{d}z}=\begin{pmatrix}\langle\alpha^1+1,\frac{w}{d}\rangle & & 0\\ & \ddots & \\ 0 & & \langle\alpha^{\mu_f}+1,\frac{w}{d}\rangle\end{pmatrix}=:M.$$

In particular,

$$\mathrm{M}_f=\big[\mathrm{e}^{2\pi\mathrm{i}M}\big].$$

Proof. Let $w'=\frac{w}{d}\in\mathbb{Q}^{n+1}$. Since, by the *Euler relation*,

$$f=\sum_{i=0}^{n}w'_iz_i\partial_{z_i}f,$$

$\kappa_f=1$ and we can choose

$$\xi'_i:=w'_iz_i$$

for $i=0,\dots,n$. Since $\kappa_f=1$, there is a canonical surjection

$$H''_f/tH''_f\cong\frac{\Omega^{n+1}_{X,x}}{f\Omega^{n+1}_{X,x}+\mathrm{d}f\wedge\mathrm{d}\Omega^{n-1}_{X,x}}\twoheadrightarrow\frac{\Omega^{n+1}_{X,x}}{\mathrm{d}f\wedge\Omega^n_{X,x}}\cong\mathcal{O}_{X,x}/\langle\partial_z f\rangle,$$

which is an isomorphism since

$$\dim_{\mathbb{C}}H''_f/tH''_f=\mu_f=\dim_{\mathbb{C}}\mathcal{O}_{X,x}/\langle\partial_z f\rangle.$$

Since

$$\begin{aligned} t\nabla_f\big[z^\alpha \mathrm{d}z\big] &= \Big[\Big(\sum_{i=0}^n \partial_{z_i}(z^\alpha w'_i z_i) - z^\alpha\Big)\mathrm{d}z\Big] \\ &= \Big[\Big(\sum_{i=0}^n (\alpha_i+1)w'_i - 1\Big)z^\alpha \mathrm{d}z\Big] \\ &= \big[\big(\langle \alpha+1, w'\rangle - 1\big)z^\alpha \mathrm{d}z\big] \end{aligned}$$

by proposition 3.12,

$$t\nabla_{f,\mathrm{e}\mathrm{d}z} = \begin{pmatrix} \langle\alpha_1+1,w'\rangle & & 0 \\ & \ddots & \\ 0 & & \langle\alpha_{\mu_f}+1,w'\rangle \end{pmatrix} - E$$

and, hence,

$$\begin{aligned} t\nabla_{f,t\mathrm{e}\mathrm{d}z} &= t(\nabla_{f,\mathrm{e}\mathrm{d}z})^{tE} \\ &= t\nabla_{f,\mathrm{e}\mathrm{d}z} + E \\ &= \begin{pmatrix} \langle\alpha_1+1,w'\rangle & & 0 \\ & \ddots & \\ 0 & & \langle\alpha_{\mu_f}+1,w'\rangle \end{pmatrix} = M. \end{aligned}$$

In particular,

$$\mathrm{M}_f = \mathrm{M}_{\nabla_f} = \mathrm{M}_{\nabla_{f,t\mathrm{e}\mathrm{d}z}} = \mathrm{M}_{\frac{M}{t}} = \big[\mathrm{e}^{2\pi\mathrm{i}M}\big]$$

by definition 3.9 and corollary 1.34. $\square$

In the general case, we use the following result of Brieskorn to compute $t^{\kappa_f}\nabla_f$ on H''_f up to arbitrarily high order.

Proposition 3.14 (Existence of $N'_f(K)$ and $N''_f(K)$)**.**
The residue maps

$$\Omega^n_{X,x} \twoheadrightarrow H'_f\ ,$$

$$\Omega^{n+1}_{X,x} \twoheadrightarrow H''_f$$

are continous with respect to the adic topologies defined by $\mathfrak{m}_{S,s}$ respectively $\mathfrak{m}_{X,x}$. In particular, for each $K \in \mathbb{N}$, there are minimal $N'_f(K), N''_f(K) \in \mathbb{N}$ with

$$\begin{aligned} \mathfrak{m}_{X,x}^{N'_f(K)}\Omega^n_{X,x} &\subset f^K\Omega^n_{X,x} + \mathrm{d}f\wedge\Omega^{n-1}_{X,x} + \mathrm{d}\Omega^{n-1}_{X,x}, \\ \mathfrak{m}_{X,x}^{N''_f(K)}\Omega^{n+1}_{X,x} &\subset f^K\Omega^{n+1}_{X,x} + \mathrm{d}f\wedge\mathrm{d}\Omega^{n-1}_{X,x}. \end{aligned}$$

Proof. E. Brieskorn [Bri70], proposition 3.3 and the following remarks. $\square$

The following corollary shows how to compute the map

$$H''_f \xrightarrow{\mathrm{jet}^{H''_f}_{K-1}(t^{\kappa_f}\nabla_f)} H''_f/t^K H''_f$$

induced by $t^{\kappa_f}\nabla_f$.

Corollary 3.15 (Computing $\mathrm{jet}^{H''_f}_{K-1}(t^{\kappa_f}\nabla_f)$)**.**
Let $K \in \mathbb{N}$ and

$$\begin{aligned} V_{K,N} := &\big(\mathrm{jet}_{N-1-|\alpha|}\, f^K z^\alpha\big)_{0\le|\alpha|<N-K\,\mathrm{ord}\, f},\\ &\big(\mathrm{jet}_{N-1}\,\partial_z f\big)_{\mathrm{ord}\, f\le N},\\ &\big(\mathrm{jet}_{N-1-|\alpha|}\big(\alpha_i z_j\partial_{z_i} f - \alpha_j z_i\partial_{z_j} f\big) z^\alpha\big)_{\substack{0\le|\alpha|<N-\mathrm{ord}\, f\\ 0\le i<j\le n}}. \end{aligned}$$

Then,

$$\langle V_{K,N}\mathrm{d}z\rangle_{\mathbb{C}}/\mathfrak{m}^N_{X,x}\Omega^{n+1}_{X,x} = \big(f^K\Omega^{n+1}_{X,x} + \mathrm{d}f\wedge\mathrm{d}\Omega^{n-1}_{X,x} + \mathfrak{m}^N_{X,x}\Omega^{n+1}_{X,x}\big)/\mathfrak{m}^N_{X,x}\Omega^{n+1}_{X,x}$$

with

$$\mathrm{codim}_{\mathbb{C}}\big(\langle V_{K,N}\mathrm{d}z\rangle_{\mathbb{C}}/\mathfrak{m}^N_{X,x}\Omega^{n+1}_{X,x} \subset \Omega^{n+1}_{X,x}/\mathfrak{m}^N_{X,x}\Omega^{n+1}_{X,x}\big) \le K\mu_f$$

and equality holds if and only if $N \ge N''_f(K)$. In this case,

$$H''_f/t^K H''_f \cong \big(\Omega^{n+1}_{X,x}/\mathfrak{m}^N_{X,x}\Omega^{n+1}_{X,x}\big)/\big(\langle V_{K,N}\mathrm{d}z\rangle_{\mathbb{C}}/\mathfrak{m}^N_{X,x}\Omega^{n+1}_{X,x}\big)$$

and $t^{\kappa_f}\nabla_f$ induces the map

$$H''_f \xrightarrow{\mathrm{jet}^{H''_f}_{K-1}(t^{\kappa_f}\nabla_f)} H''_f/t^K H''_f$$

with

$$\mathrm{jet}^{H''_f}_{K-1}(t^{\kappa_f}\nabla_f)\,[e\mathrm{d}z] = \left[\mathrm{jet}_{N-1}\left(\sum_{i=0}^n \partial_{z_i}\big(e\,\mathrm{jet}_{N-\mathrm{ord}(e\xi_i)}(u^{-1})\xi_i\big) - \kappa_f f^{\kappa_f-1}e\right)\mathrm{d}z\right].$$

Proof. Let

$$V_{K,\infty} := \big(f^K z^\alpha\big)_{0\le|\alpha|},\big(\partial_z f\big),\big(\big(\alpha_i z_j\partial_{z_i} f - \alpha_j z_i\partial_{z_j} f\big)z^\alpha\big)_{\substack{0\le|\alpha|\\ 0\le i<j\le n}}.$$

Since

$$\begin{aligned} \mathrm{d}f\wedge\mathrm{d}(z_i\mathrm{d}z_{\hat{i},\hat{j}}) &= (-1)^i\mathrm{d}f\wedge\mathrm{d}z_{\hat{j}}\\ &= (-1)^{i+j}\partial_{z_j} f\mathrm{d}z \end{aligned}$$

and

$$\begin{aligned} \mathrm{d}f\wedge\mathrm{d}(z_i z_j z^\alpha\mathrm{d}z_{\hat{i},\hat{j}}) &= \mathrm{d}f\wedge\big((-1)^i\alpha_i z_j z^\alpha\mathrm{d}z_{\hat{j}} - (-1)^j\alpha_j z_i z^\alpha\mathrm{d}z_{\hat{i}}\big)\\ &= (-1)^{i+j}\big(\alpha_i z_j\partial_{z_j} f - \alpha_j z_i\partial_{z_i} f\big)z^\alpha\mathrm{d}z, \end{aligned}$$

$$\langle V_{K,\infty}\mathrm{d}z\rangle_{\mathbb{C}} = f^K\Omega^{n+1}_{X,x} + \mathrm{d}f\wedge\mathrm{d}\Omega^{n-1}_{X,x}.$$

Clearly,

$$V_{K,\infty}\mathrm{d}z/\mathfrak{m}^N_{X,x}\Omega^{n+1}_{X,x} = V_{K,N}\mathrm{d}z/\mathfrak{m}^N_{X,x}\Omega^{n+1}_{X,x}$$

and, hence,

$$\langle V_{K,N}\mathrm{d}z\rangle_{\mathbb{C}}/\mathfrak{m}^N_{X,x}\Omega^{n+1}_{X,x} = \big(f^K\Omega^{n+1}_{X,x} + \mathrm{d}f\wedge\mathrm{d}\Omega^{n-1}_{X,x} + \mathfrak{m}^N_{X,x}\Omega^{n+1}_{X,x}\big)/\mathfrak{m}^N_{X,x}\Omega^{n+1}_{X,x}.$$

There is a canonical surjection

$$H''_f/t^K H''_f \cong \frac{\Omega^{n+1}_{X,x}}{f^K\Omega^{n+1}_{X,x}+\mathrm{d}f\wedge\mathrm{d}\Omega^{n-1}_{X,x}} \xrightarrow{\phi} \frac{\Omega^{n+1}_{X,x}/\mathfrak{m}^N_{X,x}\Omega^{n+1}_{X,x}}{(f^K\Omega^{n+1}_{X,x}+\mathrm{d}f\wedge\mathrm{d}\Omega^{n-1}_{X,x}+\mathfrak{m}^N_{X,x}\Omega^{n+1}_{X,x})/\mathfrak{m}^N_{X,x}\Omega^{n+1}_{X,x}}.$$

Hence, since $H''_f/t^K H''_f \cong \big(\mathcal{O}_{S,s}/\mathfrak{m}^K_{S,s}\mathcal{O}_{S,s}\big)^{\mu_f} \cong \mathbb{C}^{K\mu_f}$,

$$\begin{aligned}\operatorname{codim}_{\mathbb{C}}\big(\langle V_{K,N}\mathrm{d}z\rangle_{\mathbb{C}}/\mathfrak{m}^N_{X,x}\Omega^{n+1}_{X,x} \subset \Omega^{n+1}_{X,x}/\mathfrak{m}^N_{X,x}\Omega^{n+1}_{X,x}\big) &\le \dim_{\mathbb{C}}(H''_f/t^K H''_f)\\ &= K\mu_f.\end{aligned}$$

Equality holds if and only if ϕ is an isomorphism or, equivalently,

$$\mathfrak{m}^N_{X,x}\Omega^{n+1}_{X,x} \subset f^K\Omega^{n+1}_{X,x} + \mathrm{d}f\wedge\mathrm{d}\Omega^{n-1}_{X,x}.$$

By definition of $N''_f(K)$ in proposition 3.14, this is equivalent to $N \ge N''_f(K)$.

The formula for $\mathrm{jet}^{H''_f}_{K-1}\big(t^{\kappa_f}\nabla_f\big)$ follows immediately from proposition 3.12. □

The following corollary shows how to compute the connection matrix $M := t^{\kappa_f}\nabla_{f,e\mathrm{d}z}$ for a basis $e\mathrm{d}z$ of H''_f up to arbitrarily high order $K-1$.

Corollary 3.16 (Computing $\mathrm{jet}_{K-1}(t^{\kappa_f}\nabla_{f,e\mathrm{d}z})$).

If $e\mathrm{d}z$ with $e \in \mathbb{C}\{z\}^n$ represents a $\mathbb{C}$-basis of H''_f/tH''_f, then $e\mathrm{d}z$ represents an $\mathcal{O}_{S,s}$-basis of H''_f and, for each $K\in\mathbb{N}$, $P^e_K\mathrm{d}z$ with

$$P^e_K := \big(f^i e_j\big)_{\substack{i=0,\dots,K-1\\ j=1,\dots,\mu_f}}$$

represents a $\mathbb{C}$-basis of $H''_f/t^K H''_f$.

If $N \ge N''_f$, then there is a unique $C = (c^k_{i,j})^{k=0,\dots,K-1}_{i,j=1,\dots,\mu_f} \in \mathrm{Mat}(\mu_f,\mathbb{C})^K$ with

$$\begin{aligned}\mathrm{jet}_{N-1}\Bigg(\sum_{j=0}^{n}\partial_{z_j}\big(e_i\,\mathrm{jet}_{N-\mathrm{ord}(e_i\xi_j)}(u^{-1})\xi_j\big) - \kappa_f f^{\kappa_f-1}e_i\Bigg) &=\\ \sum_{j=1}^{\mu_f}\sum_{k=0}^{K-1} c^k_{i,j}\,\mathrm{jet}_{N-1}(f^k e_j) \bmod \langle V_{K,N}\rangle_{\mathbb{C}}.&\end{aligned}$$

and

$$\mathrm{jet}_{K-1}\, m_{i,j} = \sum_{k=0}^{K-1} c^k_{i,j}t^k$$

where $M = (m_{i,j})_{i,j=1,\dots,\mu_f} := t^{\kappa_f}\nabla_{f,e\mathrm{d}z}$.

Proof. The first statement follows from the Nakayama lemma and the second, from proposition 3.3.

If $N \ge N''_f$, then, by corollary 3.15,

$$H''_f/t^K H''_f \cong \big(\Omega^{n+1}_{X,x}/\mathfrak{m}^N_{X,x}\Omega^{n+1}_{X,x}\big)/\big(\langle V_{K,N}\mathrm{d}z\rangle_{\mathbb{C}}/\mathfrak{m}^N_{X,x}\Omega^{n+1}_{X,x}\big).$$

Hence, by the second statement, $(V^e_{K,N}\mathrm{d}z, V_{K,N}\mathrm{d}z)$ with

$$V^e_{K,N} := \mathrm{jet}_{N-1} P^e_K$$

represents a $\mathbb{C}$-basis of $\Omega^{n+1}_{X,x}/\mathfrak{m}^N_{X,x}\Omega^{n+1}_{X,x}$ and

$$\begin{aligned} H''_f/t^K H''_f &\cong \bigoplus_{j=1}^{\mu_f}(\mathbb{C}\{t\}/t^K\mathbb{C}\{t\})e_j \\ &\cong \bigoplus_{j=1}^{\mu_f}\bigoplus_{k=0}^{K-1}\mathbb{C}\,\mathrm{jet}_{N-1}(f^k e_j). \end{aligned}$$

Hence, the third statement follows from corollary 3.15. □

4. A Singular Procedure to Compute the Monodromy

In this section, we explain the implementation of the algorithm in the computer algebra system Singular [GPS98] and compute some examples.

Singular can compute standard bases over rings with local ordering. Using the Singular extension `pcv` in the appendix and the arithmetic for modules, Singular can do linear algebra on finite dimensional algebras. Both features are essential for the algorithm.

4.1. **Implementation.** In the following, we describe the Singular procedure `monodromy` in the Singular library `mondromy.lib` in the appendix.
The input is a polynomial f in a series ring, i.e. a ring with local ordering, defining an isolated hypersurface singularity and an optional integer *opt* where $opt = 0$ by default.
If $opt = 0$, the output is a matrix M such that $\mathrm{e}^{2\pi\mathrm{i}M}$ is a monodromy matrix of the given singularity, else, the output is a matrix M such that $\chi_{\mathrm{e}^{2\pi\mathrm{i}M}}$ is the characteristic polynomial of the monodromy of the given singularity.
First, we compute the Milnor number μ_f using `milnor` and return an empty matrix if f does not define an isolated singularity. If f does define an isolated singularity, we use `qhweight` to test if f is quasihomogenous and, if this is the case, to compute quasihomogenous weights $w \in \mathbb{N}^{n+1}$ such that

$$f = \sum_{\langle\alpha,w\rangle=d} f_\alpha z^\alpha.$$

```
int mu=milnor(f);

if(mu<=0)
{
  if(mu==0)
  {
    print("//no singularity");
  }
  else
  {
    print("//non isolated singularity");
  }

  matrix M[1][0];
  return(M);
}
else
{
  intvec w=qhweight(f);

  if(w==0)
  {
```

```
    return(nonqhmonodromy(f,mu,opt));
  }
  else
  {
    return(qhmonodromy(f,w));
  }
}
```

If f is quasihomogenous, we continue in the procedure `qhmonodromy`. We compute a monomial $\mathbb{C}$-basis $e = (z^{\alpha^i})_{i=1,\ldots,\mu_f}$ of the Milnor algebra $\mathbb{C}\{z\}/\langle\partial_z f\rangle$ of f using `kbase` and the matrix

$$M = \begin{pmatrix} \langle\alpha^1+1,\frac{w}{d}\rangle & & 0 \\ & \ddots & \\ 0 & & \langle\alpha^{\mu_f}+1,\frac{w}{d}\rangle \end{pmatrix}$$

defined in corollary 3.13 and return M.

```
ideal e=kbase(std(jacob(f)));
int mu,d=size(e),(transpose(leadexp(f))*w)[1];

matrix M[mu][mu];
int i;
for(i=mu;i>=1;i--)
{
  M[i,i]=number((transpose(leadexp(e[i])+1)*w)[1])/d;
}

return(M);
```

If f is not quasihomogenous, we continue in the procedure `nonqhmonodromy`. We compute

$$\kappa_f = \min\{\kappa | f^\kappa \in \langle\partial_z f\rangle\}$$

and ξ and u with

$$f^{\kappa_f} = \sum_{i=0}^{n} \frac{\xi_i}{u}\partial_{z_i} f$$

as in proposition 3.1.

```
list jl=jacoblift(f);
def kappa,xi,u=jl[1..3];
```

The procedure `jacoblift` computes κ_f by successively increasing κ and using `reduce` in each step to check if $f^\kappa \in \langle\partial_z f\rangle$ and ξ and u using `lift` and polynomial division.

```
proc jacoblift(poly f)
{
  ideal jf=jacob(f);
  ideal sjf=std(jf);
  int kappa=1;
  poly fkappa=f;
  while(reduce(fkappa,sjf)!=0)
  {
    kappa++;
    fkappa=fkappa*f;
  }

  vector xi=lift(jf,fkappa)[1];

  poly u=(matrix(jf)*xi)[1,1]/fkappa;

  return(list(kappa,xi,u));
}
```

The procedure `invunit` computes the inverse of a unit u in a series ring up to arbitrarily high order n using `lift` in the quotient ring of the corresponding polynomial ring by the $(n+1)$-th power of the maximal ideal.

```
proc invunit(poly u,int n)
{
  if(pcvmindeg(u)==0)
  {
    def br=basering;
    changeord("pr","dp");
    qring qr=std(maxideal(n+1));
    kill pr;

    poly v=lift(fetch(br,u),1)[1,1];

    setring br;
    return(fetch(qr,v));
  }
  else
  {
    print("//no power series unit");
    return(poly(0));
  }
}
```

In corollary 3.15, we defined

$$\begin{aligned} V_{K,N} = &\big(\operatorname{jet}_{N-1-|\alpha|} f^K z^\alpha\big)_{0\le|\alpha|<N-K\operatorname{ord} f},\\ &\big(\operatorname{jet}_{N-1}\partial_z f\big)_{\operatorname{ord} f\le N},\\ &\big(\operatorname{jet}_{N-1-|\alpha|}\big(\alpha_i z_j\partial_{z_i} f-\alpha_j z_i\partial_{z_j} f\big) z^\alpha\big)_{\substack{0\le|\alpha|<N-\operatorname{ord} f\\ 0\le i<j\le n}}\end{aligned}$$

with

$$\langle V_{K,N}\mathrm{d}z\rangle_{\mathbb{C}}\big/\mathfrak{m}_{X,x}^N\Omega_{X,x}^{n+1}=\big(f^K\Omega_{X,x}^{n+1}+\mathrm{d}f\wedge\mathrm{d}\Omega_{X,x}^{n-1}+\mathfrak{m}_{X,x}^N\Omega_{X,x}^{n+1}\big)\big/\mathfrak{m}_{X,x}^N\Omega_{X,x}^{n+1}.$$

Since we want to update $V_{K,N}$ when increasing K and N, we store

$$P^1_{K,N} := \big(f^K z^\alpha\big)_{0\le|\alpha|<N-K\operatorname{ord} f}$$

and

$$P^2_N := \big(\partial_z f\big)_{\operatorname{ord} f\le N}, \Big(\big(\alpha_i z_j\partial_{z_i} f - \alpha_j z_i\partial_{z_j} f\big) z^\alpha\Big)_{\substack{0\le|\alpha|<N-\operatorname{ord} f\\ 0\le i<j\le n}}$$

as lists of polynomials and

$$V^1_{K,N} := \big(\operatorname{jet}_{N-1-|\alpha|} f^K z^\alpha\big)_{0\le|\alpha|<N-K\operatorname{ord} f}$$

and

$$\begin{aligned} V^2_N :=& \big(\operatorname{jet}_{N-1}\partial_z f\big)_{\operatorname{ord} f\le N},\\ & \Big(\operatorname{jet}_{N-1-|\alpha|}\big(\alpha_i z_j\partial_{z_i} f - \alpha_j z_i\partial_{z_j} f\big) z^\alpha\Big)_{\substack{0\le|\alpha|<N-\operatorname{ord} f\\ 0\le i<j\le n}}\end{aligned}$$

as lists of corresponding coefficient vectors with respect to a degree ordering. Using the procedures `pcvP2CV` and `pcvCV2P` from the SINGULAR extension `pcv` in the appendix, we can convert between polynomials and corresponding coefficient vectors between given degree bounds. If we want to increase N by δN, we compute

$$\begin{aligned} \delta P^1_{K,N,\delta N} &:= P^1_{K,N+\delta N} - P^1_{K,N},\\ V^1_{K,N+\delta N} &= V^1_{K,N} + \big(\operatorname{jet}_{N+\delta N-1} - \operatorname{jet}_{N-1}\big) P^1_{K,N} + \operatorname{jet}_{N+\delta N-1}\delta P^1_{K,N,\delta N},\\ P^1_{K,N+\delta N} &= P^1_{K,N} + \delta P^1_{K,N,\delta N},\end{aligned}$$

$$\begin{aligned} \delta P^2_{N,\delta N} &:= P^2_{N+\delta N} - P^2_N,\\ V^2_{N+\delta N} &= V^2_N + \big(\operatorname{jet}_{N+\delta N-1} - \operatorname{jet}_{N-1}\big) P^2_N + \operatorname{jet}_{N+\delta N-1}\delta P^2_{N,\delta N},\\ P^2_{N+\delta N} &= P^2_N + \delta P^2_{N,\delta N}.\end{aligned}$$

```
deltaP1=getdeltaP1(f,K,N,deltaN);
V1=pcvladdl(V1,pcvp2cv(P1,N,N+deltaN))+pcvp2cv(deltaP1,0,N+deltaN);
P1=P1+deltaP1;

deltaP2=getdeltaP2(f,N,deltaN);
V2=pcvladdl(V2,pcvp2cv(P2,N,N+deltaN))+pcvp2cv(deltaP2,0,N+deltaN);
P2=P2+deltaP2;
```

The procedures `getdeltaP1` and `getdeltaP2` compute $\delta P^1_{K,N,\delta N}$ and $\delta P^2_{N,\delta N}$.

```
static proc getdeltaP1(poly f,int K,int N,int deltaN)
{
  return(pcvpmull(f^K,pcvbasis(0,N+deltaN-K*pcvmindeg(f))));
}

static proc getdeltaP2(poly f,int N,int dN)
{
  def of,jf=pcvmindeg(f),jacob(f);
  list b=pcvbasis(N-of+2,N+dN-of+2);
```

```
  list P2;
  P2[size(b)*((nvars(basering)-1)*nvars(basering))/2]=0;
  int i,j,k,l;
  intvec alpha;
  for(k,l=1,1;k<=size(b);k++)
  {
    alpha=leadexp(b[k]);
    for(i=nvars(basering)-1;i>=1;i--)
    {
      for(j=nvars(basering);j>i;j--)
      {
        P2[l]=alpha[i]*jf[j]*(b[k]/var(i))-alpha[j]*jf[i]*(b[k]/var(j));
        l++;
      }
    }
  }
  return(P2);
}
```

If we want to find $N''_f(K)$ with

$$H''_f/t^K H''_f \cong \big(\Omega^{n+1}_{X,x}/\mathfrak{m}^{N''_f(K)}_{X,x}\Omega^{n+1}_{X,x}\big)/\big(\langle V_{K,N''_f(K)}\mathrm{d}z\rangle_{\mathbb{C}}/\mathfrak{m}^{N''_f(K)}_{X,x}\Omega^{n+1}_{X,x}\big),$$

then, by corollary 3.15, we have to increase N until the inequality

$$\operatorname{codim}_{\mathbb{C}}\big(\langle V_{K,N}\mathrm{d}z\rangle_{\mathbb{C}}/\mathfrak{m}^N_{X,x}\Omega^{n+1}_{X,x}\subset\Omega^{n+1}_{X,x}/\mathfrak{m}^N_{X,x}\Omega^{n+1}_{X,x}\big)\le K\mu_f$$

becomes an equality.

```
deltaN=1;
while(codimV(V1+V2,N)<K*mu)
{
  deltaP1=getdeltaP1(f,K,N,deltaN);
  V1=pcvladdl(V1,pcvp2cv(P1,N,N+deltaN))+pcvp2cv(deltaP1,0,N+deltaN);
  P1=P1+deltaP1;

  deltaP2=getdeltaP2(f,N,deltaN);
  V2=pcvladdl(V2,pcvp2cv(P2,N,N+deltaN))+pcvp2cv(deltaP2,0,N+deltaN);
  P2=P2+deltaP2;

  N=N+deltaN;
}
```

The procedure `codimV` computes

$$\operatorname{codim}_{\mathbb{C}}\big(\langle V\rangle_{\mathbb{C}}/\mathfrak{m}^N\subset\mathbb{C}\{z\}/\mathfrak{m}^N\big)$$

using `interred` to get a $\mathbb{C}$-basis of $\langle V\rangle_{\mathbb{C}}/\mathfrak{m}^N$.

```
static proc codimV(list V,int N)
{
  int codim=pcvdim(0,N);
  if(size(V)>0)
  {
    codim=codim-ncols(interred(module(V[1..size(V)])));
  }
  return(codim);
}
```

If we want to increase K by δK, we set

$$\delta N := \delta K \operatorname{ord} f$$

and compute

$$P^1_{K+\delta K,N+\delta N} = f^{\delta K} P^1_{K,N},$$
$$V^1_{K+\delta K,N+\delta N} = \mathrm{jet}_{N+\delta N-1}\, P^1_{K+\delta K,N+\delta N},$$

and $P^2_{N+\delta N}$ and $V^2_{N+\delta N}$ as above.

```
int deltaN=deltaK*pcvmindeg(f);

list deltaP1;
P1=pcvpmull(f^deltaK,P1);
V1=pcvp2cv(P1,0,N+deltaN);
```

The procedure `incK` increases K by δK and N until $N = N''_f(K)$.

```
static proc incK(poly f,int mu,int K,int deltaK,int N,
  list e,list P1,list P2,list Pe,list V1,list V2,list Ve)
{
  int deltaN=deltaK*pcvmindeg(f);

  list deltaP1;
  P1=pcvpmull(f^deltaK,P1);
  V1=pcvp2cv(P1,0,N+deltaN);

  list deltaP2=getdeltaP2(f,N,deltaN);
  V2=pcvladdl(V2,pcvp2cv(P2,N,N+deltaN))+pcvp2cv(deltaP2,0,N+deltaN);
  P2=P2+deltaP2;

  list deltaPe=getdeltaPe(f,e,K,deltaK);
  Ve=pcvladdl(Ve,pcvp2cv(Pe,N,N+deltaN))+pcvp2cv(deltaPe,0,N+deltaN);
  Pe=Pe+deltaPe;

  K=K+deltaK;
  dbprint(printlevel-voice+2,"//K="+string(K));

  N=N+deltaN;
  dbprint(printlevel-voice+2,"//N="+string(N));

  deltaN=1;
  while(codimV(V1+V2,N)<K*mu)
  {
    deltaP1=getdeltaP1(f,K,N,deltaN);
    V1=pcvladdl(V1,pcvp2cv(P1,N,N+deltaN))+pcvp2cv(deltaP1,0,N+deltaN);
    P1=P1+deltaP1;

    deltaP2=getdeltaP2(f,N,deltaN);
    V2=pcvladdl(V2,pcvp2cv(P2,N,N+deltaN))+pcvp2cv(deltaP2,0,N+deltaN);
    P2=P2+deltaP2;

    Ve=pcvladdl(Ve,pcvp2cv(Pe,N,N+deltaN));

    N=N+deltaN;
```

```
    dbprint(printlevel-voice+2,"//N="+string(N));
  }

  return(K,N,P1,P2,Pe,V1,V2,Ve);
}
```

Since we need a $\mathbb{C}$-basis of H''_f/tH''_f to use corollary 3.16, we first increase $K=0$ by $\delta K=1$.

```
int K,N,prevN;
list e,P1,P2,Pe,V1,V2,Ve,Vnablae;

K,N,P1,P2,Pe,V1,V2,Ve=incK(f,mu,K,1,N,e,P1,P2,Pe,V1,V2,Ve);
```

Then, $K=1$ and $N=N''_f(1)$ and, hence,

$$H''_f/tH''_f \cong \left(\Omega^{n+1}_{X,x}/\mathfrak{m}^N_{X,x}\Omega^{n+1}_{X,x}\right)/\left(\langle V_{K,N}\mathrm{d}z\rangle_{\mathbb{C}}/\mathfrak{m}^N_{X,x}\Omega^{n+1}_{X,x}\right)$$

by corollary 3.15. Hence, we can compute a list of polynomials e such that $e\mathrm{d}z$ is a $\mathbb{C}$-basis of H''_f/tH''_f.

```
e=pcvcv2p(quotvs(V1+V2,N),0,N);
```

The procedure `quotV` computes a $\mathbb{C}$-basis of $\left(\mathbb{C}\{z\}/\mathfrak{m}^N\right)/\left(\langle V\rangle_{\mathbb{C}}/\mathfrak{m}^N\right)$ as a list of coefficient vectors.

```
static proc quotV(list V,int N)
{
  module Q=freemodule(pcvdim(0,N));
  if(size(V)>0)
  {
    Q=interred(reduce(std(Q),std(module(V[1..size(V)]))));
  }
  return(list(Q[1..size(Q)]));
}
```

In corollary 3.16, we defined

$$P^e_K = \left(f^i e_j\right)_{\substack{i=0,\dots,K-1,\\ j=1,\dots,\mu_f}},$$
$$V^e_{K,N} = \left(\mathrm{jet}_{N-1}\left(f^i e_j\right)\right)_{\substack{i=0,\dots,K-1.\\ j=1,\dots,\mu_f}}$$

We store P^e_K as a list of polynomials and $V^e_{K,N}$ as a list of corresponding coefficient vectors. Since $K=1$, we compute

$$P^e_1 = e,$$
$$V^e_{1,N} = \mathrm{jet}_{N-1} P^e_1.$$

```
  Pe=e;
  Ve=pcvp2cv(Pe,0,N);
```

If we want to increase K by δK and N by δN, we compute

$$\begin{aligned}
\delta P^e_{K,\delta K} &:= P^e_{K+\delta K} - P^e_K,\\
V^e_{K+\delta K,N+\delta N} &= V^e_{K,N} + \big(\mathrm{jet}_{N+\delta N-1} - \mathrm{jet}_{N-1}\big)P^e_K + \mathrm{jet}_{N+\delta N-1}\,\delta P^e_{K,\delta K},\\
P^e_{K+\delta K} &= P^e_K + \delta P^e_{K,\delta K}.
\end{aligned}$$

```
list deltaPe=getdeltaPe(f,e,K,deltaK);
Ve=pcvladdl(Ve,pcvp2cv(Pe,N,N+deltaN))+pcvp2cv(deltaPe,0,N+deltaN);
Pe=Pe+deltaPe;
```

The procedure `getdeltaPe` computes $\delta P^e_{K,\delta K}$.

```
static proc getdeltaPe(poly f,list e,int K,int deltaK)
{
  int k;
  list Pe,fke;
  for(k,fke=K,pcvpmull(f^K,e);k<K+deltaK;k,fke=k+1,pcvpmull(f,fke))
  {
    Pe=Pe+fke;
  }
  return(Pe);
}
```

If we just want to increase N by δN, we compute

$$V^e_{K+\delta K,N+\delta N} = V^e_{K,N} + \big(\mathrm{jet}_{N+\delta N-1} - \mathrm{jet}_{N-1}\big)P^e_K.$$

```
Ve=pcvladdl(Ve,pcvp2cv(Pe,N,N+deltaN));
```

P^e_K and $V^e_{K,N}$ are computed by the procedure `incK` when increasing K and N.
The procedure `nablaK` computes the map $\mathrm{jet}^{H''_f}_{K-1}(t^{\kappa_f}\nabla_f)$ using the formula

$$\mathrm{jet}^{H''_f}_{K-1}(t^{\kappa_f}\nabla_f)\,[e\mathrm{d}z] = \left[\mathrm{jet}_{N-1}\left(\sum_{i=0}^{n}\partial_{z_i}\big(e\,\mathrm{jet}_{N-\mathrm{ord}(e\xi_i)}(u^{-1})\xi_i\big) - \kappa_f f^{\kappa_f-1}e\right)\mathrm{d}z\right]$$

from corollary 3.15. More precisely, it computes the expression

$$\mathrm{jet}_{N-1}\left(\sum_{i=0}^{n}\partial_{z_i}\big(e\,\mathrm{jet}_{N-\mathrm{ord}(e\xi_i)}(u^{-1})\xi_i\big) - \kappa_f f^{\kappa_f-1}e\right)$$

in corollary 3.16.

```
static proc nablaK(poly f,int kappa,vector xi,poly u,int N,int prevN,
  list Vnablae,list e)
{
  xi=jet(xi,N);
  u=invunit(u,N);
  poly fkappa=kappa*f^(kappa-1);

  poly p,q;
  list nablae;
  int i,j;
  for(i=1;i<=size(e);i++)
  {
    for(j,p=nvars(basering),0;j>=1;j--)
    {
      q=jet(e[i]*xi[j],N);
      if(q!=0)
      {
        p=p+diff(q*jet(u,N-pcvmindeg(q)),var(j));
      }
    }
    nablae=nablae+list(p-jet(fkappa*e[i],N-1));
  }

  return(pcvladdl(Vnablae,pcvp2cv(nablae,prevN,N-prevN)));
}
```

The procedure MK computes $\mathrm{jet}_{K-1} M$ where

$$M = (m_{i,j})_{i,j=1,\dots,\mu_f} = t^{\kappa_f}\nabla_{f,\mathrm{d}z}$$

using the formula

$$\mathrm{jet}_{K-1} m_{i,j} = \sum_{k=0}^{K-1} c_{i,j}^k t^k$$

from corollary 3.16. $C = (c_{i,j}^k)_{i,j=1,\dots,\mu_f}^{k=0,\dots,K-1} \in \mathrm{Mat}(\mu_f, \mathbb{C})^K$ with

$$\mathrm{jet}_{N-1}\left(\sum_{j=0}^{n} \partial_{z_j}\left(e_i \,\mathrm{jet}_{N-\mathrm{ord}(e_i\xi_j)}(u^{-1})\xi_j\right) - \kappa_f f^{\kappa_f-1} e_i\right) = \sum_{j=1}^{\mu_f}\sum_{k=0}^{K-1} c_{i,j}^k \,\mathrm{jet}_{N-1}(f^k e_j) \bmod \langle V_{K,N}\rangle_{\mathbb{C}}.$$

as in corollary 3.16 is computed using the procedure `nablaK` and `lift`.

```
static proc MK(poly f,int mu,int kappa,vector xi,poly u,
  int K,int N,int prevN,list e,list V1,list V2,list Ve,list Vnablae)
{
  Vnablae=pcvladdl(Vnablae,pcvp2cv(nablaK(f,kappa,xi,u,N,e),prevN,N-prevN));

  list V=Ve+V1+V2;
  matrix C=lift(module(V[1..size(V)]),module(Vnablae[1..size(Vnablae)]));

  int i1,i2,j,k;
  matrix M[mu][mu];
```

```
  for(j=1;j<=mu;j++)
  {
    for(k,i2=0,1;k<K;k++)
    {
      for(i1=1;i1<=mu;i1,i2=i1+1,i2+1)
      {
        M[i1,j]=M[i1,j]+C[i2,j]*var(1)^k;
      }
    }
  }

  return(M,N,Vnablae);
}
```

If $\kappa_f = 1$, we compute the residue matrix $M_0' := M_0$.

```
matrix M=list(MK(f,mu,kappa,xi,u,K,N,prevN,e,V1,V2,Ve,Vnablae))[1];
```

If $\kappa_f > 1$, we use corollary 2.12 to transform $\frac{M}{t^{\kappa_f}}$ to a simple pole. We successively compute sets of generators of the lattices L_i defined in corollary 2.12 by

$$L_0 = t^{(\kappa_f-1)(\mu_f-1)}\mathbb{C}\{t\}^{\mu_f},$$
$$L_i = L_{i-1} + \mathrm{jet}_{(\kappa_f-1)(\mu_f-1)}\left(\frac{\mathrm{jet}_{(\kappa_f-1)i}\, M}{t^{\kappa_f-1}} L_{i-1}\right) \text{ for } i \in \mathbb{N}$$

and check in each step if $L_i = L_{i+1}$ using `reduce`. By corollary 2.12, we find $L_\infty = L_{i_\infty}$ with

$$i_\infty = \min\{i | L_i = L_{i+1}\} \leq \mu_f - 1$$

and compute $U \in \mathrm{Iso}_{\mathbb{C}\{t\}}(\mathbb{C}\{t\}^{\mu_f}, L_\infty) \subset \mathrm{GL}_{\mu_f}(K)$ with

$$\mathrm{pol}\left(\frac{M}{t^{\kappa_f}}\right)^U \leq 1$$

using `minbase`.

```
matrix U=freemodule(mu)*var(1)^((mu-1)*(kappa-1));
matrix M,prevU;
int i;
for(i=mu-1;i>=1&&size(reduce(U,std(prevU)))>0;i--)
{
  K,N,P1,P2,Pe,V1,V2,Ve=incK(f,mu,K,kappa-1,N,e,P1,P2,Pe,V1,V2,Ve);
  M,prevN,Vnablae=MK(f,mu,kappa,xi,u,K,N,prevN,e,V1,V2,Ve,Vnablae);
  prevU=U;
  U=interred(jet(module(U)+module(var(1)*diff(U,var(1)))+
    module(mdivp(M*U,var(1)^(kappa-1))),(kappa-1)*(mu-1)));
}
if(ncols(U)>nrows(U))
{
  U=minbase(U);
}
```

We multiply U by $t^{\mathrm{pol}\, U}$ to decrease $\mathrm{pol}\, U^{-1}$.

```
U=mdivp(U,var(1)^pcvmindeg(U));
```

The procedure `detadj` computes the determinant $\det U$ and adjoint matrix $\hat{U}$ of a square matrix U using `det` and `lift`.

```
proc detadj(module U)
{
  if(nrows(U)==ncols(U))
  {
    poly detU=det(U);

    if(detU==0)
    {
      print("//determinant zero");
      return(list(detU));
    }
    else
    {
      def br=basering;
      changeord("pr","dp");
      matrix U=fetch(br,U);
      poly detU=fetch(br,detU);

      matrix adjU=lift(U,detU*freemodule(nrows(U)));

      setring br;
      matrix adjU=fetch(pr,adjU);
      kill pr;
    }
  }
  else
  {
    print("//no square matrix");
    return(list());
  }

  return(list(detU,adjU));
}
```

Since we want to get the residue matrix after the transformation, we compute

$$\frac{M'}{t} := \frac{\operatorname{jet}_0\Big(\big(\frac{\det U}{t^{\operatorname{ord}\det U}}\big)^{-1}\Big)\hat{U}\big(t^{\kappa_f}\partial_t U + \operatorname{jet}_{\kappa_f+\operatorname{ord}\det U} MU\big)}{t^{\kappa_f+\operatorname{ord}\det U}}$$

such that

$$\operatorname{jet}_{-1}\frac{M'}{t} = \operatorname{jet}_{-1}\left(\left(\frac{M}{t^{\kappa_f}}\right)^U\right).$$

```
list daU=detadj(U);
poly p=var(1)^min(intvec(pcvmindeg(daU[2]),pcvmindeg(daU[1])));
daU[1]=daU[1]/p;
daU[2]=mdivp(daU[2],p);

if(K<kappa+pcvmindeg(daU[1]))
```

```
{
  K,N,P1,P2,Pe,V1,V2,Ve=
    incK(f,mu,K,kappa+pcvmindeg(daU[1])-K,N,e,P1,P2,Pe,V1,V2,Ve);
  M,prevN,Vnablae=MK(f,mu,kappa,xi,u,K,N,prevN,e,V1,V2,Ve,Vnablae);
}

M=mdivp(daU[2]*(var(1)^kappa*diff(U,var(1))+M*U),
  leadcoef(daU[1])*var(1)^(kappa+pcvmindeg(daU[1])-1));
```

Now, we have computed the residue matrix M'_0 in the case $\kappa_f = 1$ or $\kappa_f > 1$.
If *opt* $= 0$, we return M'_0 by corollary 1.33, else, we compute the maximum integer difference of eigenvalues $\delta_{M'_0}$ of M'_0.

```
list jd=jordan(M);
def eM0,bM0=jd[1..2];
int delta=mid(eM0);
```

The procedure `jordan` from the SINGULAR library `jordan.lib` in the appendix computes the eigenvalues and corresponding jordan block sizes of the constant part of a square matrix with eigenvalues in the coefficient field. We can apply it to M' since M'_0 has rational eigenvalues by proposition 3.11 and corollary 1.33. The procedure `mid` computes the maximum integer difference.

```
static proc mid(ideal l)
{
  int i,j,id;
  int mid=0;
  for(i=size(l);i>=1;i--)
  {
    for(j=i-1;j>=1;j--)
    {
      id=int(l[i]-l[j]);
      id=max(intvec(id,-id));
      mid=max(intvec(id,mid));
    }
  }
  return(mid);
}
```

If $\delta_{M'_0} = 0$, we return M'_0 by proposition 1.31.

```
matrix M0=jet(M,0);
return(M0);
```

If $\delta_{M0} > 0$, we apply the transformation from proposition 1.32 $\delta_{M'_0}$ times to make $\delta_{M'_0} = 0$. Since we want to get the residue matrix after the transformation, we first compute

$$M' := \mathrm{jet}_{\delta_{M_0}} M$$

if $\kappa_f = 1$ or

$$\frac{M'}{t} := \frac{\mathrm{jet}_{\delta_{M_0}}\left(\left(\frac{\det U}{t^{\mathrm{ord}\det U}}\right)^{-1}\right)\hat{U}\left(t^{\kappa_f}\partial_t U + \mathrm{jet}_{\kappa_f + \mathrm{ord}\det U + \delta_{M_0}} MU\right)}{t^{\kappa_f + \mathrm{ord}\det U}}$$

if $\kappa_f > 1$ such that in both cases

$$\mathrm{jet}_{-1+\delta_{M_0}} \frac{M'}{t} = \mathrm{jet}_{-1+\delta_{M_0}}\left(\left(\frac{M}{t^{\kappa_f}}\right)^U\right).$$

```
if(kappa>1)
{
  K,N,P1,P2,Pe,V1,V2,Ve=
    incK(f,mu,K,kappa+pcvmindeg(daU[1])+delta-K,N,e,P1,P2,Pe,V1,V2,Ve);
}
else
{
  K,N,P1,P2,Pe,V1,V2,Ve=incK(f,mu,K,kappa+delta-K,N,e,P1,P2,Pe,V1,V2,Ve);
}

M,prevN,Vnablae=MK(f,mu,kappa,xi,u,K,N,prevN,e,V1,V2,Ve,Vnablae);

if(kappa>1)
{
  M=mdivp(invunit(daU[1]/var(1)^pcvmindeg(daU[1]),delta)*
    daU[2]*(var(1)^kappa*diff(U,var(1))+M*U),
    var(1)^(kappa+pcvmindeg(daU[1])-1));
}

M=decmide(M,eM0,mM0);
delta--;

while(delta>0)
{
  jd=jordan(M);
  eM0,bM0=jd[1..2];
  M=decmide(M,eM0,mM0);
  delta--;
}
```

The procedure `decmide` computes the transformation from proposition 1.32 by computing the generalized eigenspaces of M'_0 using `syz`, transforming M' linearly to make

$$M' = \begin{pmatrix} {M'}^{1,1} & {M'}^{1,2} \\ {M'}^{2,1} & {M'}^{2,2} \end{pmatrix}$$

with blocks corresponding to the generalized eigenspaces of M'_0 as in proposition 1.32, and computing

$$M'' := \begin{pmatrix} {M'}^{1,1} + E & \frac{{M'}^{1,2}}{t} \\ t{M'}^{2,1} & {M'}^{2,2} \end{pmatrix}.$$

```
static proc decmide(matrix M,ideal eM0,list bM0)
{
  matrix M0=jet(M,0);

  int i,j;
  matrix U,M0e;
  matrix E=freemodule(nrows(M));
  for(i=ncols(eM0);i>=1;i--)
  {
    M0e=E;
    for(j=max(bM0[i]);j>=1;j--)
    {
      M0e=M0e*(M0-eM0[i]*E);
    }
    U=syz(M0e)+U;
  }

  list daU=detadj(U);
  daU[2]=(1/number(daU[1]))*daU[2];
  M=daU[2]*M*U;

  int k;
  intvec ideM0;
  ideM0[ncols(eM0)]=0;
  for(i=ncols(eM0);i>=1;i--)
  {
    for(j=ncols(eM0);j>=1;j--)
    {
      k=int(eM0[i]-eM0[j]);
      if(k)
      {
        if(k>0)
        {
          ideM0[i]=max(intvec(k,ideM0[i]));
        }
        else
        {
          ideM0[j]=max(intvec(-k,ideM0[j]));
        }
      }
    }
  }
  for(i,k=size(bM0),nrows(M);i>=1;i--)
  {
    for(j=sum(bM0[i]);j>=1;j--)
    {
      ideM0[k]=ideM0[i];
      k--;
    }
  }

  for(i=nrows(M);i>=1;i--)
  {
    if(!ideM0[i])
    {
      M[i,i]=M[i,i]+1;
    }
```

```
    for(j=ncols(M);j>=1;j--)
    {
      if(ideM0[i]&&!ideM0[j])
      {
        M[i,j]=M[i,j]*var(1);
      }
      else
      {
        if(!ideM0[i]&&ideM0[j])
        {
          M[i,j]=M[i,j]/var(1);
        }
      }
    }
  }

  return(M);
}
```

Finally, we return M_0'' by proposition 1.31.

```
matrix M0=jet(M,0);
return(M0);
```

In oder to keep the vector space $\Omega^{n+1}_{X,x}/\mathfrak{m}^N_{X,x}\Omega^{n+1}_{X,x}$ as small as possible, we increase K and N in steps of one. Nevertheless, the dimension

$$\dim_{\mathbb{C}} \Omega^{n+1}_{X,x}/\mathfrak{m}^N_{X,x}\Omega^{n+1}_{X,x} = \binom{N+n+1}{n+1}$$

becomes quite large in general and expressing vectors in a basis of $\Omega^{n+1}_{X,x}/\mathfrak{m}^N_{X,x}\Omega^{n+1}_{X,x}$ takes a main part of the time of the whole computation. Moreover, the expressions for ξ and u can be quite complicated depending on the choice of the polynomial defining a given type of singularity. These are the two main problems of the algorithm.

4.2. **Usage and Examples.** To demonstrate the usage of the SINGULAR procedure `monodromy`, we compute an example by Malgrange. We use procedures from the SINGULAR library `jordan.lib` in the appendix to compute the Jordan normal form of the monodromy matrix.

```
                     SINGULAR                             /
 A Computer Algebra System for Polynomial Computations   /   version 1.3.1
                                                       0<
     by: G.-M. Greuel, G. Pfister, H. Schoenemann        \   November 1998
FB Mathematik der Universitaet, D-67653 Kaiserslautern    \
> LIB "mondromy.lib";
> ring R=0,(x,y),ds;
> poly f=x2y2+x6+y6;
> matrix M=monodromy(f);
> print(M);
7/6,0,  0,0,  0,  0,0,   0,-1/2,0,  0,  0,  0,
0,  7/6,0,0,  0,  0,-1/2,0,0,   0,  0,  0,  0,
0,  0,  1,0,  0,  0,0,   0,0,   0,  0,  0,  0,
```

```
0,  0,  0,4/3,0,  0,0,   0,0,   0,  0,  0,  0,
0,  0,  0,0,  4/3,0,0,   0,0,   0,  0,  0,  0,
0,  0,  0,0,  0,  1,0,   0,0,   0,  0,  0,  0,
0,  0,  0,0,  0,  0,5/6, 0,0,   0,  0,  0,  0,
0,  0,  0,0,  0,  0,0,   1,0,   0,  0,  0,  0,
0,  0,  0,0,  0,  0,0,   0,5/6, 0,  0,  0,  0,
0,  0,  0,0,  0,  0,0,   0,0,   2/3,0,  0,  0,
0,  0,  0,0,  0,  0,0,   0,0,   0,  2/3,0,  0,
0,  0,  0,0,  0,  0,0,   0,0,   0,  0,  1,  -1/3,
0,  0,  0,0,  0,  0,0,   0,0,   0,  0,  3/4,0
> LIB "jordan.lib";
> print(jordanform(M));
1/2,1,  0,  0,  0,  0,  0,0,0,0,  0,  0,  0,
0,  1/2,0,  0,  0,  0,  0,0,0,0,  0,  0,  0,
0,  0,  2/3,0,  0,  0,  0,0,0,0,  0,  0,  0,
0,  0,  0,  2/3,0,  0,  0,0,0,0,  0,  0,  0,
0,  0,  0,  0,  5/6,0,  0,0,0,0,  0,  0,  0,
0,  0,  0,  0,  0,  5/6,0,0,0,0,  0,  0,  0,
0,  0,  0,  0,  0,  0,  1,0,0,0,  0,  0,  0,
0,  0,  0,  0,  0,  0,  0,1,0,0,  0,  0,  0,
0,  0,  0,  0,  0,  0,  0,0,1,0,  0,  0,  0,
0,  0,  0,  0,  0,  0,  0,0,0,7/6,0,  0,  0,
0,  0,  0,  0,  0,  0,  0,0,0,0,  7/6,0,  0,
0,  0,  0,  0,  0,  0,  0,0,0,0,  0,  4/3,0,
0,  0,  0,  0,  0,  0,  0,0,0,0,  0,  0,  4/3
> jordan(M);
[1]:
   _[1]=1/2
   _[2]=2/3
   _[3]=5/6
   _[4]=1
   _[5]=7/6
   _[6]=4/3
[2]:
   [1]:
      2
   [2]:
      1,1
   [3]:
      1,1
   [4]:
      1,1,1
   [5]:
      1,1
   [6]:
      1,1
```

The above example was computed on a Pentium II 350 with a memory of 128 M plus 256 M swap memory under Linux in less than one second using about 400 K of memory. Using Maple V Release 5 and the Maple V procedure `Monodromy` by P. F. M. Nacken [Nac90], the same result can be computed in about 45 seconds using about 3 M of memory.

As above, we computed the monodromy of some unimodal and bimodal singularities. The results and computation times are listed in the following tables. The first column contains the input polynomial, the second the type of singularity defined by it, the third a list of pairs of eigenvalues and corresponding Jordan block size vectors of the monodromy of this singularity, and the fourth the computation time in seconds. The computation time can depend very much on the chosen ring ordering but there seems to be no favorite one. Most of the following examples we could not compute using Nacken's procedure since it needs too much memory.

Table 1. The Monodromy of some Unimodal Singulartites

polynomial	type	eigenvalues and Jordan block sizes of monodromy	time
$z^3+y^3+xyz+x^3$	P_8	$\big(1,(1)\big)$, $\big(\frac{4}{3},(1,1,1)\big)$, $\big(\frac{5}{3},(1,1,1)\big)$, $\big(2,(1)\big)$	0
$y^4+x^2y^2+x^4$	X_9	$\big(\frac{1}{2},(1)\big)$, $\big(\frac{3}{4},(1,1)\big)$, $\big(1,(1,1,1)\big)$, $\big(\frac{5}{4},(1,1)\big)$, $\big(\frac{3}{2},(1)\big)$	0
$y^6+x^2y^2+x^3$	J_{10}	$\big(\frac{1}{2},(1)\big)$, $\big(\frac{2}{3},(1)\big)$, $\big(\frac{5}{6},(1,1)\big)$, $\big(1,(1,1)\big)$, $\big(\frac{7}{6},(1,1)\big)$, $\big(\frac{4}{3},(1)\big)$, $\big(\frac{3}{2},(1)\big)$	0
$z^5+y^4+xyz+x^3$	$T_{3,4,5}$	$\big(1,(2)\big)$, $\big(\frac{6}{5},(1)\big)$, $\big(\frac{5}{4},(1)\big)$, $\big(\frac{4}{3},(1)\big)$, $\big(\frac{7}{5},(1)\big)$, $\big(\frac{3}{2},(1)\big)$, $\big(\frac{8}{5},(1)\big)$, $\big(\frac{5}{3},(1)\big)$, $\big(\frac{7}{4},(1)\big)$, $\big(\frac{9}{5},(1)\big)$	5
$z^6+y^4+xyz+x^3$	$T_{3,4,6}$	$\big(1,(2)\big)$, $\big(\frac{7}{6},(1)\big)$, $\big(\frac{5}{4},(1)\big)$, $\big(\frac{4}{3},(1,1)\big)$, $\big(\frac{3}{2},(1,1)\big)$, $\big(\frac{5}{3},(1,1)\big)$, $\big(\frac{7}{4},(1)\big)$, $\big(\frac{11}{6},(1)\big)$	14
$z^6+y^5+xyz+x^4$	$T_{4,5,6}$	$\big(1,(2)\big)$, $\big(\frac{7}{6},(1)\big)$, $\big(\frac{6}{5},(1)\big)$, $\big(\frac{5}{4},(1)\big)$, $\big(\frac{4}{3},(1)\big)$, $\big(\frac{7}{5},(1)\big)$, $\big(\frac{3}{2},(1,1)\big)$, $\big(\frac{8}{5},(1)\big)$, $\big(\frac{5}{3},(1)\big)$, $\big(\frac{7}{4},(1)\big)$, $\big(\frac{9}{5},(1)\big)$, $\big(\frac{11}{6},(1)\big)$	10
$y^7+xy^5+x^3$	E_{12}	$\big(\frac{10}{21},(1)\big)$, $\big(\frac{11}{21},(1)\big)$, $\big(\frac{13}{21},(1)\big)$, $\big(\frac{16}{21},(1)\big)$, $\big(\frac{17}{21},(1)\big)$, $\big(\frac{19}{21},(1)\big)$, $\big(\frac{20}{21},(1)\big)$, $\big(\frac{22}{21},(1)\big)$, $\big(\frac{23}{21},(1)\big)$, $\big(\frac{25}{21},(1)\big)$, $\big(\frac{26}{21},(1)\big)$, $\big(\frac{29}{21},(1)\big)$	4

$y^8+xy^5+x^3$	E_{13}	$\left(\frac{7}{15},(1)\right)$, $\left(\frac{8}{15},(1)\right)$, $\left(\frac{3}{5},(1)\right)$, $\left(\frac{11}{15},(1)\right)$, $\left(\frac{4}{5},(1)\right)$, $\left(\frac{13}{15},(1)\right)$, $\left(\frac{14}{15},(1)\right)$, $\left(1,(1)\right)$, $\left(\frac{16}{15},(1)\right)$, $\left(\frac{17}{15},(1)\right)$, $\left(\frac{6}{5},(1)\right)$, $\left(\frac{19}{15},(1)\right)$, $\left(\frac{7}{5},(1)\right)$	13
$y^8+xy^6+x^3$	E_{14}	$\left(\frac{11}{24},(1)\right)$, $\left(\frac{13}{24},(1)\right)$, $\left(\frac{7}{12},(1)\right)$, $\left(\frac{17}{24},(1)\right)$, $\left(\frac{19}{24},(1)\right)$, $\left(\frac{5}{6},(1)\right)$, $\left(\frac{11}{12},(1)\right)$, $\left(\frac{23}{24},(1)\right)$, $\left(\frac{25}{24},(1)\right)$, $\left(\frac{13}{12},(1)\right)$, $\left(\frac{7}{6},(1)\right)$, $\left(\frac{29}{24},(1)\right)$, $\left(\frac{31}{24},(1)\right)$, $\left(\frac{17}{12},(1)\right)$	3
$y^5+xy^4+x^3y$	Z_{11}	$\left(\frac{7}{15},(1)\right)$, $\left(\frac{8}{15},(1)\right)$, $\left(\frac{2}{3},(1)\right)$, $\left(\frac{11}{15},(1)\right)$, $\left(\frac{13}{15},(1)\right)$, $\left(\frac{14}{15},(1)\right)$, $\left(1,(1)\right)$, $\left(\frac{16}{15},(1)\right)$, $\left(\frac{17}{15},(1)\right)$, $\left(\frac{19}{15},(1)\right)$, $\left(\frac{4}{3},(1)\right)$	20
$xy^4+x^2y^{32}+x^3y$	Z_{12}	$\left(-\frac{6}{11},(1)\right)$, $\left(-\frac{4}{11},(1)\right)$, $\left(-\frac{3}{11},(1)\right)$, $\left(-\frac{2}{11},(1)\right)$, $\left(-\frac{1}{11},(1)\right)$, $\left(0,(1,1)\right)$, $\left(\frac{1}{11},(1)\right)$, $\left(\frac{2}{11},(1)\right)$, $\left(\frac{3}{11},(1)\right)$, $\left(\frac{4}{11},(1)\right)$	8
$y^6+xy^5+x^3y$	Z_{13}	$\left(\frac{4}{9},(1)\right)$, $\left(\frac{5}{9},(1)\right)$, $\left(\frac{11}{18},(1)\right)$, $\left(\frac{13}{18},(1)\right)$, $\left(\frac{7}{9},(1)\right)$, $\left(\frac{8}{9},(1)\right)$, $\left(\frac{17}{18},(1)\right)$, $\left(1,(1)\right)$, $\left(\frac{19}{18},(1)\right)$, $\left(\frac{10}{9},(1)\right)$, $\left(\frac{11}{9},(1)\right)$, $\left(\frac{23}{18},(1)\right)$, $\left(\frac{25}{18},(1)\right)$	7
$y^5+x^2y^3+x^4$	W_{12}	$\left(\frac{9}{20},(1)\right)$, $\left(\frac{11}{20},(1)\right)$, $\left(\frac{13}{20},(1)\right)$, $\left(\frac{7}{10},(1)\right)$, $\left(\frac{17}{20},(1)\right)$, $\left(\frac{9}{10},(1)\right)$, $\left(\frac{19}{20},(1)\right)$, $\left(\frac{21}{20},(1)\right)$, $\left(\frac{11}{10},(1)\right)$, $\left(\frac{23}{20},(1)\right)$, $\left(\frac{13}{10},(1)\right)$, $\left(\frac{27}{20},(1)\right)$	1
$y^6+xy^4+x^4$	W_{13}	$\left(\frac{7}{16},(1)\right)$, $\left(\frac{9}{16},(1)\right)$, $\left(\frac{5}{8},(1)\right)$, $\left(\frac{11}{16},(1)\right)$, $\left(\frac{13}{16},(1)\right)$, $\left(\frac{7}{8},(1)\right)$, $\left(\frac{15}{16},(1)\right)$, $\left(1,(1)\right)$, $\left(\frac{17}{16},(1)\right)$, $\left(\frac{9}{8},(1)\right)$, $\left(\frac{19}{16},(1)\right)$, $\left(\frac{21}{16},(1)\right)$, $\left(\frac{11}{8},(1)\right)$	1

$yz^2+y^4+xy^3+x^3$	Q_{10}	$\left(\frac{23}{24},(1)\right)$, $\left(\frac{25}{24},(1)\right)$, $\left(\frac{29}{24},(1)\right)$, $\left(\frac{31}{24},(1)\right)$, $\left(\frac{4}{3},(1)\right)$, $\left(\frac{35}{24},(1)\right)$, $\left(\frac{37}{24},(1)\right)$, $\left(\frac{5}{3},(1)\right)$, $\left(\frac{41}{24},(1)\right)$, $\left(\frac{43}{24},(1)\right)$	7
$z^5+y^2z+xz^3+x^3$	Q_{11}	$\left(\frac{17}{18},(1)\right)$, $\left(\frac{19}{18},(1)\right)$, $\left(\frac{7}{6},(1)\right)$, $\left(\frac{23}{18},(1)\right)$, $\left(\frac{4}{3},(1)\right)$, $\left(\frac{25}{18},(1)\right)$, $\left(\frac{3}{2},(1)\right)$, $\left(\frac{29}{18},(1)\right)$, $\left(\frac{5}{3},(1)\right)$, $\left(\frac{31}{18},(1)\right)$, $\left(\frac{11}{6},(1)\right)$	7
$yz^2+y^5+xy^4+x^3$	Q_{12}	$\left(\frac{14}{15},(1)\right)$, $\left(\frac{16}{15},(1)\right)$, $\left(\frac{17}{15},(1)\right)$, $\left(\frac{19}{15},(1)\right)$, $\left(\frac{4}{3},(1,1)\right)$, $\left(\frac{22}{15},(1)\right)$, $\left(\frac{23}{15},(1)\right)$, $\left(\frac{5}{3},(1,1)\right)$, $\left(\frac{26}{15},(1)\right)$, $\left(\frac{28}{15},(1)\right)$	7
$y^2z+xz^2+x^3z+x^4$	S_{11}	$\left(\frac{15}{16},(1)\right)$, $\left(\frac{17}{16},(1)\right)$, $\left(\frac{19}{16},(1)\right)$, $\left(\frac{5}{4},(1)\right)$, $\left(\frac{21}{16},(1)\right)$, $\left(\frac{23}{16},(1)\right)$, $\left(\frac{3}{2},(1)\right)$, $\left(\frac{25}{16},(1)\right)$, $\left(\frac{27}{16},(1)\right)$, $\left(\frac{7}{4},(1)\right)$, $\left(\frac{29}{16},(1)\right)$	12
$z^5+y^2z+xz^3+x^2y$	S_{12}	$\left(\frac{12}{13},(1)\right)$, $\left(\frac{14}{13},(1)\right)$, $\left(\frac{15}{13},(1)\right)$, $\left(\frac{16}{13},(1)\right)$, $\left(\frac{17}{13},(1)\right)$, $\left(\frac{18}{13},(1)\right)$, $\left(\frac{19}{13},(1)\right)$, $\left(\frac{20}{13},(1)\right)$, $\left(\frac{21}{13},(1)\right)$, $\left(\frac{22}{13},(1)\right)$, $\left(\frac{23}{13},(1)\right)$, $\left(\frac{24}{13},(1)\right)$	6
$z^4+y^3+xyz^2+x^3$	U_{12}	$\left(\frac{11}{12},(1)\right)$, $\left(\frac{13}{12},(1)\right)$, $\left(\frac{7}{6},(1)\right)$, $\left(\frac{5}{4},(1,1)\right)$, $\left(\frac{17}{12},(1)\right)$, $\left(\frac{3}{2},(1,1)\right)$, $\left(\frac{19}{12},(1)\right)$, $\left(\frac{7}{4},(1,1)\right)$, $\left(\frac{11}{6},(1)\right)$	3

TABLE 2. The Monodromy of some Bimodal Singulartites

polynomial	type	eigenvalues and Jordan block sizes of monodromy	time
$y^{10}+x^2y^3+x^3$	$J_{3,1}$	$\left(\frac{4}{9},(1)\right)$, $\left(\frac{9}{20},(1)\right)$, $\left(\frac{11}{20},(1)\right)$, $\left(\frac{5}{9},(1)\right)$, $\left(\frac{13}{20},(1)\right)$, $\left(\frac{3}{4},(1)\right)$, $\left(\frac{7}{9},(1)\right)$, $\left(\frac{17}{20},(1)\right)$, $\left(\frac{8}{9},(1)\right)$, $\left(\frac{19}{20},(1)\right)$, $\left(1,(1)\right)$, $\left(\frac{21}{20},(1)\right)$, $\left(\frac{10}{9},(1)\right)$, $\left(\frac{23}{20},(1)\right)$, $\left(\frac{11}{9},(1)\right)$, $\left(\frac{5}{4},(1)\right)$, $\left(\frac{27}{20},(1)\right)$	18

$y^{11}+x^2y^3+x^3$	$J_{3,2}$	$\left(\frac{4}{9},(1)\right)$, $\left(\frac{5}{11},(1)\right)$, $\left(\frac{6}{11},(1)\right)$, $\left(\frac{5}{9},(1)\right)$, $\left(\frac{7}{11},(1)\right)$, $\left(\frac{8}{11},(1)\right)$, $\left(\frac{7}{9},(1)\right)$, $\left(\frac{9}{11},(1)\right)$, $\left(\frac{8}{9},(1)\right)$, $\left(\frac{10}{11},(1)\right)$, $\left(1,(1,1)\right)$, $\left(\frac{12}{11},(1)\right)$, $\left(\frac{10}{9},(1)\right)$, $\left(\frac{13}{11},(1)\right)$, $\left(\frac{11}{9},(1)\right)$, $\left(\frac{14}{11},(1)\right)$, $\left(\frac{15}{11},(1)\right)$	54
$y^8+x^2y^3+x^3y$	$Z_{1,1}$	$\left(\frac{3}{7},(1)\right)$, $\left(\frac{7}{16},(1)\right)$, $\left(\frac{9}{16},(1)\right)$, $\left(\frac{4}{7},(1)\right)$, $\left(\frac{11}{16},(1)\right)$, $\left(\frac{5}{7},(1)\right)$, $\left(\frac{13}{16},(1)\right)$, $\left(\frac{6}{7},(1)\right)$, $\left(\frac{15}{16},(1)\right)$, $\left(1,(1,1)\right)$, $\left(\frac{17}{16},(1)\right)$, $\left(\frac{8}{7},(1)\right)$, $\left(\frac{19}{16},(1)\right)$, $\left(\frac{9}{7},(1)\right)$, $\left(\frac{21}{16},(1)\right)$	10
$y^9+x^2y^3+x^3y$	$Z_{1,2}$	$\left(\frac{3}{7},(1)\right)$, $\left(\frac{4}{9},(1)\right)$, $\left(\frac{5}{9},(1)\right)$, $\left(\frac{4}{7},(1)\right)$, $\left(\frac{2}{3},(1)\right)$, $\left(\frac{5}{7},(1)\right)$, $\left(\frac{7}{9},(1)\right)$, $\left(\frac{6}{7},(1)\right)$, $\left(\frac{8}{9},(1)\right)$, $\left(1,(1,1,1)\right)$, $\left(\frac{10}{9},(1)\right)$, $\left(\frac{8}{7},(1)\right)$, $\left(\frac{11}{9},(1)\right)$, $\left(\frac{9}{7},(1)\right)$, $\left(\frac{4}{3},(1)\right)$	7
$y^6+x^2y^3+x^4$	$W_{1,0}$	$\left(\frac{5}{12},(1)\right)$, $\left(\frac{7}{12},(1)\right)$, $\left(\frac{2}{3},(1)\right)$, $\left(\frac{3}{4},(1)\right)$, $\left(\frac{5}{6},(1)\right)$, $\left(\frac{11}{12},(1,1)\right)$, $\left(1,(1)\right)$, $\left(\frac{13}{12},(1,1)\right)$, $\left(\frac{7}{6},(1)\right)$, $\left(\frac{5}{4},(1)\right)$, $\left(\frac{4}{3},(1)\right)$, $\left(\frac{17}{12},(1)\right)$, $\left(\frac{19}{12},(1)\right)$	0
$y^7+x^2y^3+x^4$	$W_{1,1}$	$\left(\frac{5}{12},(1)\right)$, $\left(\frac{3}{7},(1)\right)$, $\left(\frac{4}{7},(1)\right)$, $\left(\frac{7}{12},(1)\right)$, $\left(\frac{2}{3},(1)\right)$, $\left(\frac{5}{7},(1)\right)$, $\left(\frac{5}{6},(1)\right)$, $\left(\frac{6}{7},(1)\right)$, $\left(\frac{11}{12},(1)\right)$, $\left(1,(1,1)\right)$, $\left(\frac{13}{12},(1)\right)$, $\left(\frac{8}{7},(1)\right)$, $\left(\frac{7}{6},(1)\right)$, $\left(\frac{9}{7},(1)\right)$, $\left(\frac{4}{3},(1)\right)$	4
$y^8+x^2y^3+x^4$	$W_{1,2}$	$\left(\frac{5}{12},(1)\right)$, $\left(\frac{7}{16},(1)\right)$, $\left(\frac{9}{16},(1)\right)$, $\left(\frac{7}{12},(1)\right)$, $\left(\frac{2}{3},(1)\right)$, $\left(\frac{11}{16},(1)\right)$, $\left(\frac{13}{16},(1)\right)$, $\left(\frac{5}{6},(1)\right)$, $\left(\frac{11}{12},(1)\right)$, $\left(\frac{15}{16},(1)\right)$, $\left(1,(1)\right)$, $\left(\frac{17}{16},(1)\right)$, $\left(\frac{13}{12},(1)\right)$, $\left(\frac{7}{6},(1)\right)$, $\left(\frac{19}{16},(1)\right)$, $\left(\frac{21}{16},(1)\right)$, $\left(\frac{4}{3},(1)\right)$	5

$x^4+2x^2y^3+xy^5+y^6$	$W^{\#}_{1,1}$	$\big(\frac{5}{12},(1)\big)$, $\big(\frac{11}{26},(1)\big)$, $\big(\frac{15}{26},(1)\big)$, $\big(\frac{7}{12},(1)\big)$, $\big(\frac{17}{26},(1)\big)$, $\big(\frac{19}{26},(1)\big)$, $\big(\frac{21}{26},(1)\big)$, $\big(\frac{23}{26},(1)\big)$, $\big(\frac{11}{12},(1)\big)$, $\big(\frac{25}{26},(1)\big)$, $\big(\frac{27}{26},(1)\big)$, $\big(\frac{13}{12},(1)\big)$, $\big(\frac{29}{26},(1)\big)$, $\big(\frac{31}{26},(1)\big)$, $\big(\frac{33}{26},(1)\big)$, $\big(\frac{35}{26},(1)\big)$	2797
$x^4+2x^2y^3+x^2y^4+y^6$	$W^{\#}_{1,2}$	$\big(\frac{5}{12},(1)\big)$, $\big(\frac{3}{7},(1)\big)$, $\big(\frac{4}{7},(1)\big)$, $\big(\frac{7}{12},(1)\big)$, $\big(\frac{9}{14},(1)\big)$, $\big(\frac{5}{7},(1)\big)$, $\big(\frac{11}{14},(1)\big)$, $\big(\frac{6}{7},(1)\big)$, $\big(\frac{11}{12},(1)\big)$, $\big(\frac{13}{14},(1)\big)$, $\big(1,(1)\big)$, $\big(\frac{15}{14},(1)\big)$, $\big(\frac{13}{12},(1)\big)$, $\big(\frac{8}{7},(1)\big)$, $\big(\frac{17}{14},(1)\big)$, $\big(\frac{9}{7},(1)\big)$, $\big(\frac{19}{14},(1)\big)$	426
$x^4+2x^2y^3+y^6+xy^6$	$W^{\#}_{1,3}$	$\big(\frac{5}{12},(1)\big)$, $\big(\frac{13}{30},(1)\big)$, $\big(\frac{17}{30},(1)\big)$, $\big(\frac{7}{12},(1)\big)$, $\big(\frac{19}{30},(1)\big)$, $\big(\frac{7}{10},(1)\big)$, $\big(\frac{23}{30},(1)\big)$, $\big(\frac{5}{6},(1)\big)$, $\big(\frac{9}{10},(1)\big)$, $\big(\frac{11}{12},(1)\big)$, $\big(\frac{29}{30},(1)\big)$, $\big(\frac{31}{30},(1)\big)$, $\big(\frac{13}{12},(1)\big)$, $\big(\frac{11}{10},(1)\big)$, $\big(\frac{7}{6},(1)\big)$, $\big(\frac{37}{30},(1)\big)$, $\big(\frac{13}{10},(1)\big)$, $\big(\frac{41}{30},(1)\big)$	688
$yz^2+xy^4+x^2y^2+x^3$	$Q_{2,0}$	$\big(\frac{11}{12},(1)\big)$, $\big(\frac{13}{12},(1)\big)$, $\big(\frac{5}{4},(1,1)\big)$, $\big(\frac{4}{3},(1)\big)$, $\big(\frac{17}{12},(1,1)\big)$, $\big(\frac{19}{12},(1,1)\big)$, $\big(\frac{5}{3},(1)\big)$, $\big(\frac{7}{4},(1,1)\big)$, $\big(\frac{23}{12},(1)\big)$, $\big(\frac{25}{12},(1)\big)$	0
$yz^2+y^7+x^2y^2+x^3$	$Q_{2,1}$	$\big(\frac{11}{12},(1)\big)$, $\big(\frac{13}{14},(1)\big)$, $\big(\frac{15}{14},(1)\big)$, $\big(\frac{13}{12},(1)\big)$, $\big(\frac{17}{14},(1)\big)$, $\big(\frac{5}{4},(1)\big)$, $\big(\frac{4}{3},(1)\big)$, $\big(\frac{19}{14},(1)\big)$, $\big(\frac{17}{12},(1)\big)$, $\big(\frac{3}{2},(1)\big)$, $\big(\frac{19}{12},(1)\big)$, $\big(\frac{23}{14},(1)\big)$, $\big(\frac{5}{3},(1)\big)$, $\big(\frac{7}{4},(1)\big)$, $\big(\frac{25}{14},(1)\big)$	122
$yz^2+y^8+x^2y^2+x^3$	$Q_{2,2}$	$\big(\frac{11}{12},(1)\big)$, $\big(\frac{15}{16},(1)\big)$, $\big(\frac{17}{16},(1)\big)$, $\big(\frac{13}{12},(1)\big)$, $\big(\frac{19}{16},(1)\big)$, $\big(\frac{5}{4},(1)\big)$, $\big(\frac{21}{16},(1)\big)$, $\big(\frac{4}{3},(1)\big)$, $\big(\frac{17}{12},(1)\big)$, $\big(\frac{23}{16},(1)\big)$, $\big(\frac{25}{16},(1)\big)$, $\big(\frac{19}{12},(1)\big)$, $\big(\frac{5}{3},(1)\big)$, $\big(\frac{27}{16},(1)\big)$, $\big(\frac{7}{4},(1)\big)$, $\big(\frac{29}{16},(1)\big)$	289

$yz^2+y^3z+y^5+x^2z$	$S_{1,0}$	$\left(\frac{9}{10},(1)\right)$, $\left(\frac{11}{10},(1)\right)$, $\left(\frac{6}{5},(1)\right)$, $\left(\frac{13}{10},(1,1)\right)$, $\left(\frac{7}{5},(1)\right)$, $\left(\frac{3}{2},(1,1)\right)$, $\left(\frac{8}{5},(1)\right)$, $\left(\frac{17}{10},(1,1)\right)$, $\left(\frac{9}{5},(1)\right)$, $\left(\frac{19}{10},(1)\right)$, $\left(\frac{21}{10},(1)\right)$	0
$yz^2+y^6+x^2z+x^2y^2$	$S_{1,1}$	$\left(\frac{9}{10},(1)\right)$, $\left(\frac{11}{12},(1)\right)$, $\left(\frac{13}{12},(1)\right)$, $\left(\frac{11}{10},(1)\right)$, $\left(\frac{6}{5},(1)\right)$, $\left(\frac{5}{4},(1)\right)$, $\left(\frac{13}{10},(1)\right)$, $\left(\frac{7}{5},(1)\right)$, $\left(\frac{17}{12},(1)\right)$, $\left(\frac{3}{2},(1)\right)$, $\left(\frac{19}{12},(1)\right)$, $\left(\frac{8}{5},(1)\right)$, $\left(\frac{17}{10},(1)\right)$, $\left(\frac{7}{4},(1)\right)$, $\left(\frac{9}{5},(1)\right)$	55
$yz^2+y^7+x^2z+x^2y^2$	$S_{1,2}$	$\left(\frac{9}{10},(1)\right)$, $\left(\frac{13}{14},(1)\right)$, $\left(\frac{15}{14},(1)\right)$, $\left(\frac{11}{10},(1)\right)$, $\left(\frac{6}{5},(1)\right)$, $\left(\frac{17}{14},(1)\right)$, $\left(\frac{13}{10},(1)\right)$, $\left(\frac{19}{14},(1)\right)$, $\left(\frac{7}{5},(1)\right)$, $\left(\frac{3}{2},(1,1)\right)$, $\left(\frac{8}{5},(1)\right)$, $\left(\frac{23}{14},(1)\right)$, $\left(\frac{17}{10},(1)\right)$, $\left(\frac{25}{14},(1)\right)$, $\left(\frac{9}{5},(1)\right)$	155
$x^2z+yz^2+y^3z+xy^4$	$S^{\#}_{1,1}$	$\left(\frac{9}{10},(1)\right)$, $\left(\frac{10}{11},(1)\right)$, $\left(\frac{12}{11},(1)\right)$, $\left(\frac{11}{10},(1)\right)$, $\left(\frac{13}{11},(1)\right)$, $\left(\frac{14}{11},(1)\right)$, $\left(\frac{13}{10},(1)\right)$, $\left(\frac{15}{11},(1)\right)$, $\left(\frac{16}{11},(1)\right)$, $\left(\frac{3}{2},(1)\right)$, $\left(\frac{17}{11},(1)\right)$, $\left(\frac{18}{11},(1)\right)$, $\left(\frac{17}{10},(1)\right)$, $\left(\frac{19}{11},(1)\right)$, $\left(\frac{20}{11},(1)\right)$	91
$x^2z+yz^2+y^3z+x^2y^3$	$S^{\#}_{1,2}$	$\left(\frac{9}{10},(1)\right)$, $\left(\frac{11}{12},(1)\right)$, $\left(\frac{13}{12},(1)\right)$, $\left(\frac{11}{10},(1)\right)$, $\left(\frac{7}{6},(1)\right)$, $\left(\frac{5}{4},(1)\right)$, $\left(\frac{13}{10},(1)\right)$, $\left(\frac{4}{3},(1)\right)$, $\left(\frac{17}{12},(1)\right)$, $\left(\frac{3}{2},(1,1)\right)$, $\left(\frac{19}{12},(1)\right)$, $\left(\frac{5}{3},(1)\right)$, $\left(\frac{17}{10},(1)\right)$, $\left(\frac{7}{4},(1)\right)$, $\left(\frac{11}{6},(1)\right)$	628
$x^2z+yz^2+y^3z+xy^5$	$S^{\#}_{1,3}$	$\left(\frac{9}{10},(1)\right)$, $\left(\frac{12}{13},(1)\right)$, $\left(\frac{14}{13},(1)\right)$, $\left(\frac{11}{10},(1)\right)$, $\left(\frac{15}{13},(1)\right)$, $\left(\frac{16}{13},(1)\right)$, $\left(\frac{13}{10},(1)\right)$, $\left(\frac{17}{13},(1)\right)$, $\left(\frac{18}{13},(1)\right)$, $\left(\frac{19}{13},(1)\right)$, $\left(\frac{3}{2},(1)\right)$, $\left(\frac{20}{13},(1)\right)$, $\left(\frac{21}{13},(1)\right)$, $\left(\frac{22}{13},(1)\right)$, $\left(\frac{17}{10},(1)\right)$, $\left(\frac{23}{13},(1)\right)$, $\left(\frac{24}{13},(1)\right)$	161

$y^3z+xz^2+xy^3+x^3$	$U_{1,0}$	$\left(\frac{8}{9},(1)\right)$, $\left(\frac{10}{9},(1)\right)$, $\left(\frac{11}{9},(1,1)\right)$, $\left(\frac{4}{3},(1)\right)$, $\left(\frac{13}{9},(1,1)\right)$, $\left(\frac{14}{9},(1,1)\right)$, $\left(\frac{5}{3},(1)\right)$, $\left(\frac{16}{9},(1,1)\right)$, $\left(\frac{17}{9},(1)\right)$, $\left(\frac{19}{9},(1)\right)$	0
$y^2z^2+xz^2+xy^3+x^3$	$U_{1,1}$	$\left(\frac{8}{9},(1)\right)$, $\left(\frac{9}{10},(1)\right)$, $\left(\frac{11}{10},(1)\right)$, $\left(\frac{10}{9},(1)\right)$, $\left(\frac{6}{5},(1)\right)$, $\left(\frac{11}{9},(1)\right)$, $\left(\frac{13}{10},(1)\right)$, $\left(\frac{7}{5},(1)\right)$, $\left(\frac{13}{9},(1)\right)$, $\left(\frac{3}{2},(1)\right)$, $\left(\frac{14}{9},(1)\right)$, $\left(\frac{8}{5},(1)\right)$, $\left(\frac{17}{10},(1)\right)$, $\left(\frac{16}{9},(1)\right)$, $\left(\frac{9}{5},(1)\right)$	22
$y^4z+xz^2+xy^3+x^3$	$U_{1,2}$	$\left(\frac{8}{9},(1)\right)$, $\left(\frac{10}{11},(1)\right)$, $\left(\frac{12}{11},(1)\right)$, $\left(\frac{10}{9},(1)\right)$, $\left(\frac{13}{11},(1)\right)$, $\left(\frac{11}{9},(1)\right)$, $\left(\frac{14}{11},(1)\right)$, $\left(\frac{15}{11},(1)\right)$, $\left(\frac{13}{9},(1)\right)$, $\left(\frac{16}{11},(1)\right)$, $\left(\frac{17}{11},(1)\right)$, $\left(\frac{14}{9},(1)\right)$, $\left(\frac{18}{11},(1)\right)$, $\left(\frac{19}{11},(1)\right)$, $\left(\frac{16}{9},(1)\right)$, $\left(\frac{20}{11},(1)\right)$	23
$y^3z^2+xz^2+xy^3+x^3$	$U_{1,3}$	$\left(\frac{8}{9},(1)\right)$, $\left(\frac{11}{12},(1)\right)$, $\left(\frac{13}{12},(1)\right)$, $\left(\frac{10}{9},(1)\right)$, $\left(\frac{7}{6},(1)\right)$, $\left(\frac{11}{9},(1)\right)$, $\left(\frac{5}{4},(1)\right)$, $\left(\frac{4}{3},(1)\right)$, $\left(\frac{17}{12},(1)\right)$, $\left(\frac{13}{9},(1)\right)$, $\left(\frac{3}{2},(1)\right)$, $\left(\frac{14}{9},(1)\right)$, $\left(\frac{19}{12},(1)\right)$, $\left(\frac{5}{3},(1)\right)$, $\left(\frac{7}{4},(1)\right)$, $\left(\frac{16}{9},(1)\right)$, $\left(\frac{11}{6},(1)\right)$	50
$y^5z+xz^2+xy^3+x^3$	$U_{1,4}$	$\left(\frac{8}{9},(1)\right)$, $\left(\frac{12}{13},(1)\right)$, $\left(\frac{14}{13},(1)\right)$, $\left(\frac{10}{9},(1)\right)$, $\left(\frac{15}{13},(1)\right)$, $\left(\frac{11}{9},(1)\right)$, $\left(\frac{16}{13},(1)\right)$, $\left(\frac{17}{13},(1)\right)$, $\left(\frac{18}{13},(1)\right)$, $\left(\frac{13}{9},(1)\right)$, $\left(\frac{19}{13},(1)\right)$, $\left(\frac{20}{13},(1)\right)$, $\left(\frac{14}{9},(1)\right)$, $\left(\frac{21}{13},(1)\right)$, $\left(\frac{22}{13},(1)\right)$, $\left(\frac{23}{13},(1)\right)$, $\left(\frac{16}{9},(1)\right)$, $\left(\frac{24}{13},(1)\right)$	64
$y^{10}+xy^7+x^3$	E_{18}	$\left(\frac{13}{30},(1)\right)$, $\left(\frac{7}{15},(1)\right)$, $\left(\frac{8}{15},(1)\right)$, $\left(\frac{17}{30},(1)\right)$, $\left(\frac{19}{30},(1)\right)$, $\left(\frac{11}{15},(1)\right)$, $\left(\frac{23}{30},(1)\right)$, $\left(\frac{5}{6},(1)\right)$, $\left(\frac{13}{15},(1)\right)$, $\left(\frac{14}{15},(1)\right)$, $\left(\frac{29}{30},(1)\right)$, $\left(\frac{31}{30},(1)\right)$, $\left(\frac{16}{15},(1)\right)$, $\left(\frac{17}{15},(1)\right)$, $\left(\frac{7}{6},(1)\right)$, $\left(\frac{37}{30},(1)\right)$, $\left(\frac{19}{15},(1)\right)$, $\left(\frac{41}{30},(1)\right)$	21

$y^{11}+xy^7+x^3$	E_{19}	$\left(\frac{3}{7},(1)\right)$, $\left(\frac{10}{21},(1)\right)$, $\left(\frac{11}{21},(1)\right)$, $\left(\frac{4}{7},(1)\right)$, $\left(\frac{13}{21},(1)\right)$, $\left(\frac{5}{7},(1)\right)$, $\left(\frac{16}{21},(1)\right)$, $\left(\frac{17}{21},(1)\right)$, $\left(\frac{6}{7},(1)\right)$, $\left(\frac{19}{21},(1)\right)$, $\left(\frac{20}{21},(1)\right)$, $\left(1,(1)\right)$, $\left(\frac{22}{21},(1)\right)$, $\left(\frac{23}{21},(1)\right)$, $\left(\frac{8}{7},(1)\right)$, $\left(\frac{25}{21},(1)\right)$, $\left(\frac{26}{21},(1)\right)$, $\left(\frac{9}{7},(1)\right)$, $\left(\frac{29}{21},(1)\right)$	24
$y^{11}+xy^8+x^3$	E_{20}	$\left(\frac{14}{33},(1)\right)$, $\left(\frac{16}{33},(1)\right)$, $\left(\frac{17}{33},(1)\right)$, $\left(\frac{19}{33},(1)\right)$, $\left(\frac{20}{33},(1)\right)$, $\left(\frac{23}{33},(1)\right)$, $\left(\frac{25}{33},(1)\right)$, $\left(\frac{26}{33},(1)\right)$, $\left(\frac{28}{33},(1)\right)$, $\left(\frac{29}{33},(1)\right)$, $\left(\frac{31}{33},(1)\right)$, $\left(\frac{32}{33},(1)\right)$, $\left(\frac{34}{33},(1)\right)$, $\left(\frac{35}{33},(1)\right)$, $\left(\frac{37}{33},(1)\right)$, $\left(\frac{38}{33},(1)\right)$, $\left(\frac{40}{33},(1)\right)$, $\left(\frac{41}{33},(1)\right)$, $\left(\frac{43}{33},(1)\right)$, $\left(\frac{46}{33},(1)\right)$	30
$y^8+xy^6+x^3y$	Z_{17}	$\left(\frac{5}{12},(1)\right)$, $\left(\frac{11}{24},(1)\right)$, $\left(\frac{13}{24},(1)\right)$, $\left(\frac{7}{12},(1)\right)$, $\left(\frac{2}{3},(1)\right)$, $\left(\frac{17}{24},(1)\right)$, $\left(\frac{19}{24},(1)\right)$, $\left(\frac{5}{6},(1)\right)$, $\left(\frac{11}{12},(1)\right)$, $\left(\frac{23}{24},(1)\right)$, $\left(1,(1)\right)$, $\left(\frac{25}{24},(1)\right)$, $\left(\frac{13}{12},(1)\right)$, $\left(\frac{7}{6},(1)\right)$, $\left(\frac{29}{24},(1)\right)$, $\left(\frac{31}{24},(1)\right)$, $\left(\frac{4}{3},(1)\right)$	16
$y^9+xy^6+x^3y$	Z_{18}	$\left(\frac{7}{17},(1)\right)$, $\left(\frac{8}{17},(1)\right)$, $\left(\frac{9}{17},(1)\right)$, $\left(\frac{10}{17},(1)\right)$, $\left(\frac{11}{17},(1)\right)$, $\left(\frac{12}{17},(1)\right)$, $\left(\frac{13}{17},(1)\right)$, $\left(\frac{14}{17},(1)\right)$, $\left(\frac{15}{17},(1)\right)$, $\left(\frac{16}{17},(1)\right)$, $\left(1,(1,1)\right)$, $\left(\frac{18}{17},(1)\right)$, $\left(\frac{19}{17},(1)\right)$, $\left(\frac{20}{17},(1)\right)$, $\left(\frac{21}{17},(1)\right)$, $\left(\frac{22}{17},(1)\right)$, $\left(\frac{23}{17},(1)\right)$	17
$y^9+xy^7+x^3y$	Z_{19}	$\left(\frac{11}{27},(1)\right)$, $\left(\frac{13}{27},(1)\right)$, $\left(\frac{14}{27},(1)\right)$, $\left(\frac{16}{27},(1)\right)$, $\left(\frac{17}{27},(1)\right)$, $\left(\frac{19}{27},(1)\right)$, $\left(\frac{20}{27},(1)\right)$, $\left(\frac{22}{27},(1)\right)$, $\left(\frac{23}{27},(1)\right)$, $\left(\frac{25}{27},(1)\right)$, $\left(\frac{26}{27},(1)\right)$, $\left(1,(1)\right)$, $\left(\frac{28}{27},(1)\right)$, $\left(\frac{29}{27},(1)\right)$, $\left(\frac{31}{27},(1)\right)$, $\left(\frac{32}{27},(1)\right)$, $\left(\frac{34}{27},(1)\right)$, $\left(\frac{35}{27},(1)\right)$, $\left(\frac{37}{27},(1)\right)$	7

$y^7 + xy^5 + x^4$	W_{17}	$\left(\frac{2}{5},(1)\right)$, $\left(\frac{9}{20},(1)\right)$, $\left(\frac{11}{20},(1)\right)$, $\left(\frac{3}{5},(1)\right)$, $\left(\frac{13}{20},(1)\right)$, $\left(\frac{7}{10},(1)\right)$, $\left(\frac{4}{5},(1)\right)$, $\left(\frac{17}{20},(1)\right)$, $\left(\frac{9}{10},(1)\right)$, $\left(\frac{19}{20},(1)\right)$, $\left(1,(1)\right)$, $\left(\frac{21}{20},(1)\right)$, $\left(\frac{11}{10},(1)\right)$, $\left(\frac{23}{20},(1)\right)$, $\left(\frac{6}{5},(1)\right)$, $\left(\frac{13}{10},(1)\right)$, $\left(\frac{27}{20},(1)\right)$	10
$y^7 + x^2y^4 + x^4$	W_{18}	$\left(\frac{11}{28},(1)\right)$, $\left(\frac{13}{28},(1)\right)$, $\left(\frac{15}{28},(1)\right)$, $\left(\frac{17}{28},(1)\right)$, $\left(\frac{9}{14},(1)\right)$, $\left(\frac{19}{28},(1)\right)$, $\left(\frac{11}{14},(1)\right)$, $\left(\frac{23}{28},(1)\right)$, $\left(\frac{25}{28},(1)\right)$, $\left(\frac{13}{14},(1)\right)$, $\left(\frac{27}{28},(1)\right)$, $\left(\frac{29}{28},(1)\right)$, $\left(\frac{15}{14},(1)\right)$, $\left(\frac{31}{28},(1)\right)$, $\left(\frac{33}{28},(1)\right)$, $\left(\frac{17}{14},(1)\right)$, $\left(\frac{37}{28},(1)\right)$, $\left(\frac{19}{14},(1)\right)$	9
$yz^2 + y^7 + xy^5 + x^3$	Q_{16}	$\left(\frac{19}{21},(1)\right)$, $\left(\frac{20}{21},(1)\right)$, $\left(\frac{22}{21},(1)\right)$, $\left(\frac{23}{21},(1)\right)$, $\left(\frac{25}{21},(1)\right)$, $\left(\frac{26}{21},(1)\right)$, $\left(\frac{4}{3},(1,1)\right)$, $\left(\frac{29}{21},(1)\right)$, $\left(\frac{31}{21},(1)\right)$, $\left(\frac{32}{21},(1)\right)$, $\left(\frac{34}{21},(1)\right)$, $\left(\frac{5}{3},(1,1)\right)$, $\left(\frac{37}{21},(1)\right)$, $\left(\frac{38}{21},(1)\right)$	87
$yz^2 + y^8 + xy^6 + x^3$	Q_{18}	$\left(\frac{43}{48},(1)\right)$, $\left(\frac{47}{48},(1)\right)$, $\left(\frac{49}{48},(1)\right)$, $\left(\frac{53}{48},(1)\right)$, $\left(\frac{55}{48},(1)\right)$, $\left(\frac{59}{48},(1)\right)$, $\left(\frac{61}{48},(1)\right)$, $\left(\frac{4}{3},(1)\right)$, $\left(\frac{65}{48},(1)\right)$, $\left(\frac{67}{48},(1)\right)$, $\left(\frac{71}{48},(1)\right)$, $\left(\frac{73}{48},(1)\right)$, $\left(\frac{77}{48},(1)\right)$, $\left(\frac{79}{48},(1)\right)$, $\left(\frac{5}{3},(1)\right)$, $\left(\frac{83}{48},(1)\right)$, $\left(\frac{85}{48},(1)\right)$, $\left(\frac{89}{48},(1)\right)$	227
$x^2z + yz^2 + xy^4 + y^6$	S_{16}	$\left(\frac{15}{17},(1)\right)$, $\left(\frac{16}{17},(1)\right)$, $\left(\frac{18}{17},(1)\right)$, $\left(\frac{19}{17},(1)\right)$, $\left(\frac{20}{17},(1)\right)$, $\left(\frac{21}{17},(1)\right)$, $\left(\frac{22}{17},(1)\right)$, $\left(\frac{23}{17},(1)\right)$, $\left(\frac{24}{17},(1)\right)$, $\left(\frac{25}{17},(1)\right)$, $\left(\frac{26}{17},(1)\right)$, $\left(\frac{27}{17},(1)\right)$, $\left(\frac{28}{17},(1)\right)$, $\left(\frac{29}{17},(1)\right)$, $\left(\frac{30}{17},(1)\right)$, $\left(\frac{31}{17},(1)\right)$	84
$x^2z + yz^2 + y^4z + y^6$	S_{17}	$\left(\frac{7}{8},(1)\right)$, $\left(\frac{23}{24},(1)\right)$, $\left(\frac{25}{24},(1)\right)$, $\left(\frac{9}{8},(1)\right)$, $\left(\frac{7}{6},(1)\right)$, $\left(\frac{29}{24},(1)\right)$, $\left(\frac{31}{24},(1)\right)$, $\left(\frac{4}{3},(1)\right)$, $\left(\frac{11}{8},(1)\right)$, $\left(\frac{35}{24},(1)\right)$, $\left(\frac{3}{2},(1)\right)$, $\left(\frac{37}{24},(1)\right)$, $\left(\frac{13}{8},(1)\right)$, $\left(\frac{5}{3},(1)\right)$, $\left(\frac{41}{24},(1)\right)$, $\left(\frac{43}{24},(1)\right)$, $\left(\frac{11}{6},(1)\right)$	75

$y^5+xz^2+x^2y^2+x^3$	U_{16}	$\left(\frac{13}{15},(1)\right)$, $\left(\frac{14}{15},(1)\right)$, $\left(\frac{16}{15},(1)\right)$, $\left(\frac{17}{15},(1)\right)$, $\left(\frac{6}{5},(1,1)\right)$, $\left(\frac{19}{15},(1)\right)$, $\left(\frac{7}{5},(1,1)\right)$, $\left(\frac{22}{15},(1)\right)$, $\left(\frac{23}{15},(1)\right)$, $\left(\frac{8}{5},(1,1)\right)$, $\left(\frac{26}{15},(1)\right)$, $\left(\frac{9}{5},(1,1)\right)$	31

Appendix

4.3. The Singular Library `mondromy.lib`. The Singular library `mondromy.lib` provides procedures to compute the monodromy of an isolated hypersurface singularity.

```
///////////////////////////////////////////////////////////////////////////////

version="$Id: mondromy.lib,v 1.4 1999/06/22 15:47:35 mschulze Exp $";
info="
LIBRARY: mondromy.lib  PROCEDURES TO COMPUTE THE MONODROMY OF A SINGULARITY
AUTHOR:  Mathias Schulze, email: mschulze@mathematik.uni-kl.de

 invunit(u,n);         series inverse of polynomial u up to order n
 detadj(U);            determinant and adjoint matrix of square matrix U
 jacoblift(f);         lifts f^kappa in jacob(f) with minimal kappa
 monodromy(f[,opt]);   monodromy of isolated hypersurface singularity f
 H''basis(f);          basis of Brieskorn lattice H''
";

LIB "ring.lib";
LIB "sing.lib";
LIB "jordan.lib";
///////////////////////////////////////////////////////////////////////////////

static proc pcvladdl(list l1,list l2)
{
  return(system("pcvLAddL",l1,l2));
}

static proc pcvpmull(poly p,list l)
{
  return(system("pcvPMulL",p,l));
}

static proc pcvmindeg(list #)
{
  return(system("pcvMinDeg",#[1]));
}

static proc pcvp2cv(list l,int i0,int i1)
{
  return(system("pcvP2CV",l,i0,i1));
}

static proc pcvcv2p(list l,int i0,int i1)
{
  return(system("pcvCV2P",l,i0,i1));
}

static proc pcvdim(int i0,int i1)
{
  return(system("pcvDim",i0,i1));
}

static proc pcvbasis(int i0,int i1)
{
  return(system("pcvBasis",i0,i1));
```

```
}

static proc pcvinit()
{
  if(system("with","DynamicLoading"))
  {
    pcvladdl=Pcv::LAddL;
    pcvpmull=Pcv::PMulL;
    pcvmindeg=Pcv::MinDeg;
    pcvp2cv=Pcv::P2CV;
    pcvcv2p=Pcv::CV2P;
    pcvdim=Pcv::Dim;
    pcvbasis=Pcv::Basis;
  }
}
///////////////////////////////////////////////////////////////////////////////

static proc min(intvec v)
{
  int m=v[1];
  int i;
  for(i=2;i<=size(v);i++)
  {
    if(m>v[i])
    {
      m=v[i];
    }
  }
  return(m);
}
///////////////////////////////////////////////////////////////////////////////

static proc max(intvec v)
{
  int m=v[1];
  int i;
  for(i=2;i<=size(v);i++)
  {
    if(m<v[i])
    {
      m=v[i];
    }
  }
  return(m);
}
///////////////////////////////////////////////////////////////////////////////

static proc mdivp(matrix m,poly p)
{
  int i,j;
  for(i=nrows(m);i>=1;i--)
  {
    for(j=ncols(m);j>=1;j--)
    {
      m[i,j]=m[i,j]/p;
    }
  }
```

```
  return(m);
}
///////////////////////////////////////////////////////////////////////////////

proc codimV(list V,int N)
{
  int codim=pcvdim(0,N);
  if(size(V)>0)
  {
    dbprint(printlevel-voice+2,"//vector space dimension: "+string(codim));
    dbprint(printlevel-voice+2,
      "//number of subspace generators: "+string(size(V)));
    int t=timer;
    codim=codim-ncols(interred(module(V[1..size(V)])));
    dbprint(printlevel-voice+2,"//codimension: "+string(codim));
  }
  return(codim);
}
///////////////////////////////////////////////////////////////////////////////

proc quotV(list V,int N)
{
  module Q=freemodule(pcvdim(0,N));
  if(size(V)>0)
  {
    dbprint(printlevel-voice+2,"//vector space dimension: "+string(nrows(Q)));
    dbprint(printlevel-voice+2,
      "//number of subspace generators: "+string(size(V)));
    int t=timer;
    Q=interred(reduce(std(Q),std(module(V[1..size(V)]))));
  }
  return(list(Q[1..size(Q)]));
}
///////////////////////////////////////////////////////////////////////////////

proc invunit(poly u,int n)
"USAGE:   invunit(u,n); u poly, n int
ASSUME:  The polynomial u is a series unit.
RETURN:  The procedure returns the series inverse of u up to order n
         or a zero polynomial if u is no series unit.
DISPLAY: The procedure displays comments if printlevel>=1.
EXAMPLE: example invunit; shows an example.
"
{
  if(pcvmindeg(u)==0)
  {
    def br=basering;
    changeord("pr","dp");
    qring qr=std(maxideal(n+1));
    kill pr;

    dbprint(printlevel-voice+2,"//computing inverse...");
    int t=timer;
    poly v=lift(fetch(br,u),1)[1,1];
    dbprint(printlevel-voice+2,"//...inverse computed ["+string(timer-t)+
      " secs, "+string((memory(1)+1023)/1024)+" K]");
```

```
    setring br;
    return(fetch(qr,v));
  }
  else
  {
    print("//no series unit");
    return(poly(0));
  }
}
example
{ "EXAMPLE:"; echo=2;
  ring R=0,x,dp;
  invunit(1+x,10);
}
///////////////////////////////////////////////////////////////////////////////

proc detadj(module U)
"USAGE:   detadj(U); U matrix
ASSUME:  U is a square matrix with non zero determinant.
RETURN:  The procedure returns a list with at most 2 entries.
         If U is not a sqaure matrix, the list is empty.
         If U is a sqaure matrix, then the first entry is the determinant of U.
         If U is a square matrix and the determinant of U not zero,
         then the second entry is the adjoint matrix of U.
DISPLAY: The procedure displays comments if printlevel>=1.
EXAMPLE: example detadj; shows an example.
"
{
  if(nrows(U)==ncols(U))
  {
    dbprint(printlevel-voice+2,"//computing determinant...");
    int t=timer;
    poly detU=det(U);
    dbprint(printlevel-voice+2,"//...determinant computed ["+string(timer-t)+
      " secs, "+string((memory(1)+1023)/1024)+" K]");

    if(detU==0)
    {
      print("//determinant zero");
      return(list(detU));
    }
    else
    {
      def br=basering;
      changeord("pr","dp");
      matrix U=fetch(br,U);
      poly detU=fetch(br,detU);

      dbprint(printlevel-voice+2,"//computing adjoint matrix...");
      t=timer;
      matrix adjU=lift(U,detU*freemodule(nrows(U)));
      dbprint(printlevel-voice+2,"//...adjoint matrix computed ["
        +string(timer-t)+" secs, "+string((memory(1)+1023)/1024)+" K]");

      setring br;
      matrix adjU=fetch(pr,adjU);
      kill pr;
```

```
    }
  }
  else
  {
    print("//no square matrix");
    return(list());
  }

  return(list(detU,adjU));
}
example
{ "EXAMPLE:"; echo=2;
  ring R=0,x,dp;
  matrix U[2][2]=1,1+x,1+x2,1+x3;
  list daU=detadj(U);
  daU[1];
  print(daU[2]);
}
///////////////////////////////////////////////////////////////////////////////

proc jacoblift(poly f)
"USAGE:   jacoblift(f); f poly
ASSUME:  The polynomial f in a series ring (local ordering) defines
         an isolated hypersurface singularity.
RETURN:  The procedure returns a list with entries kappa, xi, u of type
         int, vector, poly such that kappa is minimal with f^kappa in jacob(f),
         u is a unit, and u*f^kappa=(matrix(jacob(f))*xi)[1,1].
DISPLAY: The procedure displays comments if printlevel>=1.
EXAMPLE: example jacoblift; shows an example.
"
{
  dbprint(printlevel-voice+2,"//computing kappa...");
  int t=timer;
  ideal jf=jacob(f);
  ideal sjf=std(jf);
  int kappa=1;
  poly fkappa=f;
  while(reduce(fkappa,sjf)!=0)
  {
    dbprint(printlevel-voice+2,"//kappa="+string(kappa));
    kappa++;
    fkappa=fkappa*f;
  }
  dbprint(printlevel-voice+2,"//kappa="+string(kappa));
  dbprint(printlevel-voice+2,"//...kappa computed ["+string(timer-t)+" secs, "
    +string((memory(1)+1023)/1024)+" K]");

  dbprint(printlevel-voice+2,"//computing xi...");
  t=timer;
  vector xi=lift(jf,fkappa)[1];
  dbprint(printlevel-voice+2,"//...xi computed ["+string(timer-t)+" secs, "
    +string((memory(1)+1023)/1024)+" K]");

  dbprint(printlevel-voice+2,"//computing u...");
  t=timer;
  poly u=(matrix(jf)*xi)[1,1]/fkappa;
  dbprint(printlevel-voice+2,"//...u computed ["+string(timer-t)+" secs, "
```

```
    +string((memory(1)+1023)/1024)+" K]");

  return(list(kappa,xi,u));
}
example
{ "EXAMPLE:"; echo=2;
  ring R=0,(x,y),ds;
  poly f=x2y2+x6+y6;
  jacoblift(f);
}
///////////////////////////////////////////////////////////////////////////////

static proc getdeltaP1(poly f,int K,int N,int dN)
{
  return(pcvpmull(f^K,pcvbasis(0,N+dN-K*pcvmindeg(f))));
}
///////////////////////////////////////////////////////////////////////////////

static proc getdeltaP2(poly f,int N,int dN)
{
  def of,jf=pcvmindeg(f),jacob(f);
  list b=pcvbasis(N-of+2,N+dN-of+2);
  list P2;
  P2[size(b)*((nvars(basering)-1)*nvars(basering))/2]=0;
  int i,j,k,l;
  intvec alpha;
  for(k,l=1,1;k<=size(b);k++)
  {
    alpha=leadexp(b[k]);
    for(i=nvars(basering)-1;i>=1;i--)
    {
      for(j=nvars(basering);j>i;j--)
      {
        P2[l]=alpha[i]*jf[j]*(b[k]/var(i))-alpha[j]*jf[i]*(b[k]/var(j));
        l++;
      }
    }
  }
  return(P2);
}
///////////////////////////////////////////////////////////////////////////////

static proc getdeltaPe(poly f,list e,int K,int dK)
{
  int k;
  list Pe,fke;
  for(k,fke=K,pcvpmull(f^K,e);k<K+dK;k,fke=k+1,pcvpmull(f,fke))
  {
    Pe=Pe+fke;
  }
  return(Pe);
}
///////////////////////////////////////////////////////////////////////////////

static proc incK(poly f,int mu,int K,int deltaK,int N,
  list e,list P1,list P2,list Pe,list V1,list V2,list Ve)
{
```

```
  int deltaN=deltaK*pcvmindeg(f);

  list deltaP1;
  P1=pcvpmull(f^deltaK,P1);
  V1=pcvp2cv(P1,0,N+deltaN);

  list deltaP2=getdeltaP2(f,N,deltaN);
  V2=pcvladdl(V2,pcvp2cv(P2,N,N+deltaN))+pcvp2cv(deltaP2,0,N+deltaN);
  P2=P2+deltaP2;

  list deltaPe=getdeltaPe(f,e,K,deltaK);
  Ve=pcvladdl(Ve,pcvp2cv(Pe,N,N+deltaN))+pcvp2cv(deltaPe,0,N+deltaN);
  Pe=Pe+deltaPe;

  K=K+deltaK;
  dbprint(printlevel-voice+2,"//K="+string(K));

  N=N+deltaN;
  dbprint(printlevel-voice+2,"//N="+string(N));

  deltaN=1;
  dbprint(printlevel-voice+2,"//computing codimension of");
  dbprint(printlevel-voice+2,"//df^dOmega^(n-1)+f^K*Omega^(n+1) in "
    +"Omega^(n+1) mod m^N*Omega^(n+1)...");
  int t=timer;
  while(codimV(V1+V2,N)<K*mu)
  {
    dbprint(printlevel-voice+2,"//...codimension computed ["+string(timer-t)
      +" secs, "+string((memory(1)+1023)/1024)+" K]");

    deltaP1=getdeltaP1(f,K,N,deltaN);
    V1=pcvladdl(V1,pcvp2cv(P1,N,N+deltaN))+pcvp2cv(deltaP1,0,N+deltaN);
    P1=P1+deltaP1;

    deltaP2=getdeltaP2(f,N,deltaN);
    V2=pcvladdl(V2,pcvp2cv(P2,N,N+deltaN))+pcvp2cv(deltaP2,0,N+deltaN);
    P2=P2+deltaP2;

    Ve=pcvladdl(Ve,pcvp2cv(Pe,N,N+deltaN));

    N=N+deltaN;
    dbprint(printlevel-voice+2,"//N="+string(N));

    dbprint(printlevel-voice+2,"//computing codimension of");
    dbprint(printlevel-voice+2,"//df^dOmega^(n-1)+f^K*Omega^(n+1) in "
      +"Omega^(n+1) mod m^N*Omega^(n+1)...");
    t=timer;
  }
  dbprint(printlevel-voice+2,"//...codimension computed ["+string(timer-t)
    +" secs, "+string((memory(1)+1023)/1024)+" K]");

  return(K,N,P1,P2,Pe,V1,V2,Ve);
}
///////////////////////////////////////////////////////////////////////////////

static proc nablaK(poly f,int kappa,vector xi,poly u,int N,int prevN,
  list Vnablae,list e)
```

```
{
  xi=jet(xi,N);
  u=invunit(u,N);
  poly fkappa=kappa*f^(kappa-1);

  poly p,q;
  list nablae;
  int i,j;
  for(i=1;i<=size(e);i++)
  {
    for(j,p=nvars(basering),0;j>=1;j--)
    {
      q=jet(e[i]*xi[j],N);
      if(q!=0)
      {
        p=p+diff(q*jet(u,N-pcvmindeg(q)),var(j));
      }
    }
    nablae=nablae+list(p-jet(fkappa*e[i],N-1));
  }

  return(pcvladdl(Vnablae,pcvp2cv(nablae,prevN,N-prevN)));
}
///////////////////////////////////////////////////////////////////////////////

static proc MK(poly f,int mu,int kappa,vector xi,poly u,
  int K,int N,int prevN,list e,list V1,list V2,list Ve,list Vnablae)
{
  dbprint(printlevel-voice+2,"//computing nabla(e)...");
  int t=timer;
  Vnablae=nablaK(f,kappa,xi,u,N,prevN,Vnablae,e);
  dbprint(printlevel-voice+2,"//...nabla(e) computed ["+string(timer-t)
    +" secs, "+string((memory(1)+1023)/1024)+" K]");

  dbprint(printlevel-voice+2,
    "//lifting nabla(e) to C-basis of H''/t^KH''...");
  list V=Ve+V1+V2;
  module W=module(V[1..size(V)]);
  dbprint(printlevel-voice+2,"//vector space dimension: "+string(nrows(W)));
  dbprint(printlevel-voice+2,"//number of generators: "+string(ncols(W)));
  t=timer;
  matrix C=lift(W,module(Vnablae[1..size(Vnablae)]));
  dbprint(printlevel-voice+2,"//...nabla(e) lifted ["+string(timer-t)
    +" secs, "+string((memory(1)+1023)/1024)+" K]");

  dbprint(printlevel-voice+2,"//computing e-lift of nabla(e)...");
  t=timer;
  int i1,i2,j,k;
  matrix M[mu][mu];
  for(j=1;j<=mu;j++)
  {
    for(k,i2=0,1;k<K;k++)
    {
      for(i1=1;i1<=mu;i1,i2=i1+1,i2+1)
      {
        M[i1,j]=M[i1,j]+C[i2,j]*var(1)^k;
      }
```

```
    }
  }
  dbprint(printlevel-voice+2,"//...e-lift of nabla(e) computed ["
    +string(timer-t)+" secs, "+string((memory(1)+1023)/1024)+" K]");

  return(M,N,Vnablae);
}
///////////////////////////////////////////////////////////////////////////////

static proc mid(ideal l)
{
  int i,j,id;
  int mid=0;
  for(i=size(l);i>=1;i--)
  {
    for(j=i-1;j>=1;j--)
    {
      id=int(l[i]-l[j]);
      id=max(intvec(id,-id));
      mid=max(intvec(id,mid));
    }
  }
  return(mid);
}
///////////////////////////////////////////////////////////////////////////////

static proc decmide(matrix M,ideal eM0,list bM0)
{
  matrix M0=jet(M,0);

  dbprint(printlevel-voice+2,
    "//computing basis U of generalized eigenspaces of M0...");
  int t=timer;
  int i,j;
  matrix U,M0e;
  matrix E=freemodule(nrows(M));
  for(i=ncols(eM0);i>=1;i--)
  {
    M0e=E;
    for(j=max(bM0[i]);j>=1;j--)
    {
      M0e=M0e*(M0-eM0[i]*E);
    }
    U=syz(M0e)+U;
  }
  dbprint(printlevel-voice+2,"//...U computed ["+string(timer-t)+" secs, "
    +string((memory(1)+1023)/1024)+" K]");

  dbprint(printlevel-voice+2,"//transforming M to U...");
  t=timer;
  list daU=detadj(U);
  daU[2]=(1/number(daU[1]))*daU[2];
  M=daU[2]*M*U;
  dbprint(printlevel-voice+2,"//...M transformed ["+string(timer-t)+" secs, "
    +string((memory(1)+1023)/1024)+" K]");

  dbprint(printlevel-voice+2,
```

```
  "//computing integer differences of eigenvalues of M0...");
t=timer;
int k;
intvec ideM0;
ideM0[ncols(eM0)]=0;
for(i=ncols(eM0);i>=1;i--)
{
  for(j=ncols(eM0);j>=1;j--)
  {
    k=int(eM0[i]-eM0[j]);
    if(k)
    {
      if(k>0)
      {
        ideM0[i]=max(intvec(k,ideM0[i]));
      }
      else
      {
        ideM0[j]=max(intvec(-k,ideM0[j]));
      }
    }
  }
}
for(i,k=size(bM0),nrows(M);i>=1;i--)
{
  for(j=sum(bM0[i]);j>=1;j--)
  {
    ideM0[k]=ideM0[i];
    k--;
  }
}
dbprint(printlevel-voice+2,
  "//...integer differences of eigenvalues of M0 computed ["+string(timer-t)
  +" secs, "+string((memory(1)+1023)/1024)+" K]");

dbprint(printlevel-voice+2,"//transforming M...");
t=timer;
for(i=nrows(M);i>=1;i--)
{
  if(!ideM0[i])
  {
    M[i,i]=M[i,i]+1;
  }
  for(j=ncols(M);j>=1;j--)
  {
    if(ideM0[i]&&!ideM0[j])
    {
      M[i,j]=M[i,j]*var(1);
    }
    else
    {
      if(!ideM0[i]&&ideM0[j])
      {
        M[i,j]=M[i,j]/var(1);
      }
    }
  }
```

```
  }
  dbprint(printlevel-voice+2,"//...M transformed ["+string(timer-t)+" secs, "
    +string((memory(1)+1023)/1024)+" K]");

  return(M);
}
///////////////////////////////////////////////////////////////////////////////

static proc nonqhmonodromy(poly f,int mu,int opt)
{
  pcvinit();

  dbprint(printlevel-voice+2,"//computing kappa, xi and u with "+
    "u*f^kappa=(matrix(jacob(f))*xi)[1,1]...");
  list jl=jacoblift(f);
  def kappa,xi,u=jl[1..3];
  dbprint(printlevel-voice+2,"//...kappa, xi and u computed");
  dbprint(printlevel-voice+2,"//kappa="+string(kappa));
  if(kappa==1)
  {
    dbprint(printlevel-voice+2,
      "//f quasihomogenous with respect to suitable coordinates");
  }
  else
  {
    dbprint(printlevel-voice+2,
      "//f not quasihomogenous for any choice of coordinates");
  }
  dbprint(printlevel-voice+2,"//xi=");
  dbprint(printlevel-voice+2,xi);
  dbprint(printlevel-voice+2,"//u="+string(u));

  int K,N,prevN;
  list e,P1,P2,Pe,V1,V2,Ve,Vnablae;

  dbprint(printlevel-voice+2,"//increasing K and N...");
  K,N,P1,P2,Pe,V1,V2,Ve=incK(f,mu,K,1,N,e,P1,P2,Pe,V1,V2,Ve);
  dbprint(printlevel-voice+2,"//...K and N increased");

  dbprint(printlevel-voice+2,"//computing C{f}-basis e of Brieskorn lattice "
    +"H''=Omega^(n+1)/df^dOmega^(n-1)...");
  int t=timer;
  e=pcvcv2p(quotV(V1+V2,N),0,N);
  dbprint(printlevel-voice+2,"//...e computed ["+string(timer-t)+" secs, "
    +string((memory(1)+1023)/1024)+" K]");

  dbprint(printlevel-voice+2,"//e=");
  dbprint(printlevel-voice+2,e);

  Pe=e;
  Ve=pcvp2cv(Pe,0,N);

  if(kappa==1)
  {
    dbprint(printlevel-voice+2,
      "//computing 0-jet M of e-matrix of t*nabla...");
    matrix M=list(MK(f,mu,kappa,xi,u,K,N,prevN,e,V1,V2,Ve,Vnablae))[1];
```

```
  dbprint(printlevel-voice+2,"//...M computed");
}
else
{
  dbprint(printlevel-voice+2,
    "//computing transformation matrix U to simple pole...");

  dbprint(printlevel-voice+2,"//computing t*nabla-stable lattice...");
  matrix M,prevU;
  matrix U=freemodule(mu)*var(1)^((mu-1)*(kappa-1));
  int i;
  dbprint(printlevel-voice+2,"//comparing with previous lattice...");
  t=timer;
  for(i=mu-1;i>=1&&size(reduce(U,std(prevU)))>0;i--)
  {
    dbprint(printlevel-voice+2,"//...compared with previous lattice ["
      +string(timer-t)+" secs, "+string((memory(1)+1023)/1024)+" K]");

    dbprint(printlevel-voice+2,"//increasing K and N...");
    K,N,P1,P2,Pe,V1,V2,Ve=incK(f,mu,K,kappa-1,N,e,P1,P2,Pe,V1,V2,Ve);
    dbprint(printlevel-voice+2,"//...K and N increased");

    dbprint(printlevel-voice+2,
      "//computing (K-1)-jet M of e-matrix of t^kappa*nabla...");
    M,prevN,Vnablae=MK(f,mu,kappa,xi,u,K,N,prevN,e,V1,V2,Ve,Vnablae);
    dbprint(printlevel-voice+2,"//...M computed");

    prevU=U;

    dbprint(printlevel-voice+2,"//enlarging lattice...");
    t=timer;
    U=interred(jet(module(U)+module(var(1)*diff(U,var(1)))+
      module(mdivp(M*U,var(1)^(kappa-1))),(kappa-1)*(mu-1)));
    dbprint(printlevel-voice+2,"//...lattice enlarged ["+string(timer-t)
      +" secs, "+string((memory(1)+1023)/1024)+" K]");

    dbprint(printlevel-voice+2,"//comparing with previous lattice...");
    t=timer;
  }
  dbprint(printlevel-voice+2,"//...compared with previous lattice ["
    +string(timer-t)+" secs, "+string((memory(1)+1023)/1024)+" K]");
  dbprint(printlevel-voice+2,"//...t*nabla-stable lattice computed");

  if(ncols(U)>nrows(U))
  {
    dbprint(printlevel-voice+2,
      "//computing C{f}-basis of t*nabla-stable lattice...");
    t=timer;
    U=minbase(U);
    dbprint(printlevel-voice+2,
      "//...C{f}-basis of t*nabla-stable lattice computed ["+string(timer-t)
      +" secs, "+string((memory(1)+1023)/1024)+" K]");
  }

  U=mdivp(U,var(1)^pcvmindeg(U));

  dbprint(printlevel-voice+2,"//...U computed");
```

```
  dbprint(printlevel-voice+2,
    "//computing determinant and adjoint matrix of U...");
  list daU=detadj(U);
  poly p=var(1)^min(intvec(pcvmindeg(daU[2]),pcvmindeg(daU[1])));
  daU[1]=daU[1]/p;
  daU[2]=mdivp(daU[2],p);
  dbprint(printlevel-voice+2,
    "//...determinant and adjoint matrix of U computed");

  if(K<kappa+pcvmindeg(daU[1]))
  {
    dbprint(printlevel-voice+2,"//increasing K and N...");
    K,N,P1,P2,Pe,V1,V2,Ve=
      incK(f,mu,K,kappa+pcvmindeg(daU[1])-K,N,e,P1,P2,Pe,V1,V2,Ve);
    dbprint(printlevel-voice+2,"//...K and N increased");

    dbprint(printlevel-voice+2,"//computing M...");
    M,prevN,Vnablae=MK(f,mu,kappa,xi,u,K,N,prevN,e,V1,V2,Ve,Vnablae);
    dbprint(printlevel-voice+2,"//...M computed");
  }

  dbprint(printlevel-voice+2,"//transforming M/t^kappa to simple pole...");
  t=timer;
  M=mdivp(daU[2]*(var(1)^kappa*diff(U,var(1))+M*U),
    leadcoef(daU[1])*var(1)^(kappa+pcvmindeg(daU[1])-1));
  dbprint(printlevel-voice+2,"//...M/t^kappa transformed to simple pole ["
    +string(timer-t)+" secs, "+string((memory(1)+1023)/1024)+" K]");
}

if(opt==0)
{
  dbprint(printlevel-voice+2,
    "//computing maximal integer difference delta of eigenvalues of M0...");
  t=timer;
  list jd=jordan(M);
  def eM0,bM0=jd[1..2];
  int delta=mid(eM0);
  dbprint(printlevel-voice+2,"//...delta computed ["+string(timer-t)
    +" secs, "+string((memory(1)+1023)/1024)+" K]");

  dbprint(printlevel-voice+2,"//delta="+string(delta));

  if(delta>0)
  {
    dbprint(printlevel-voice+2,"//increasing K and N...");
    if(kappa==1)
    {
      K,N,P1,P2,Pe,V1,V2,Ve=incK(f,mu,K,1+delta-K,N,e,P1,P2,Pe,V1,V2,Ve);
    }
    else
    {
      K,N,P1,P2,Pe,V1,V2,Ve=
        incK(f,mu,K,kappa+pcvmindeg(daU[1])+delta-K,N,e,P1,P2,Pe,V1,V2,Ve);
    }
    dbprint(printlevel-voice+2,"//...K and N increased");
```

```
      dbprint(printlevel-voice+2,"//computing M...");
      M,prevN,Vnablae=MK(f,mu,kappa,xi,u,K,N,prevN,e,V1,V2,Ve,Vnablae);
      dbprint(printlevel-voice+2,"//...M computed");

      if(kappa>1)
      {
        dbprint(printlevel-voice+2,
          "//transforming M/t^kappa to simple pole...");
        t=timer;
        M=mdivp(invunit(daU[1]/var(1)^pcvmindeg(daU[1]),delta)*
          daU[2]*(var(1)^kappa*diff(U,var(1))+M*U),
          var(1)^(kappa+pcvmindeg(daU[1])-1));
        dbprint(printlevel-voice+2,
          "//...M/t^kappa transformed to simple pole ["+string(timer-t)
          +" secs, "+string((memory(1)+1023)/1024)+" K]");
      }

      dbprint(printlevel-voice+2,"//decreasing delta...");
      M=decmide(M,eM0,bM0);
      delta--;
      dbprint(printlevel-voice+2,"//delta="+string(delta));

      while(delta>0)
      {
        jd=jordan(M);
        eM0,bM0=jd[1..2];
        M=decmide(M,eM0,bM0);
        delta--;
        dbprint(printlevel-voice+2,"//delta="+string(delta));
      }
      dbprint(printlevel-voice+2,"//...delta decreased");
    }
  }

  dbprint(printlevel-voice+2,"//computing 0-jet M0 of M...");
  matrix M0=jet(M,0);
  dbprint(printlevel-voice+2,"//...M0 computed");

  return(M0);
}
///////////////////////////////////////////////////////////////////////////////

static proc qhmonodromy(poly f,intvec w)
{
  dbprint(printlevel-voice+2,"//computing basis e of Milnor algebra...");
  int t=timer;
  ideal e=kbase(std(jacob(f)));
  dbprint(printlevel-voice+2,"//...e computed ["+string(timer-t)+" secs, "
    +string((memory(1)+1023)/1024)+" K]");

  dbprint(printlevel-voice+2,
    "//computing Milnor number mu and quasihomogeneous degree d...");
  int mu,d=size(e),(transpose(leadexp(f))*w)[1];
  dbprint(printlevel-voice+2,"...mu and d computed");

  dbprint(printlevel-voice+2,"//computing te-matrix M of t*nabla...");
  matrix M[mu][mu];
```

```
  int i;
  for(i=mu;i>=1;i--)
  {
    M[i,i]=number((transpose(leadexp(e[i])+1)*w)[1])/d;
  }
  dbprint(printlevel-voice+2,"//...M computed");

  return(M);
}
///////////////////////////////////////////////////////////////////////////////

proc monodromy(poly f, list #)
"USAGE:   monodromy(f[,opt]); f poly, opt int
ASSUME:  The polynomial f in a series ring (local ordering) defines
         an isolated hypersurface singularity.
RETURN:  The procedure returns a residue matrix M of the meromorphic
         Gauss-Manin connection of the singularity defined by f
         or an empty matrix if the assumptions are not fulfilled.
         If opt=0 (default), exp(-2*pi*i*M) is a monodromy matrix of f,
         else, only the characteristic polynomial of exp(-2*pi*i*M) coincides
         with the characteristic polynomial of the monodromy of f.
THEORY:  The procedure uses an algorithm by Brieskorn (See E. Brieskorn,
         manuscipta math. 2 (1970), 103-161) to compute a connection matrix of
         the meromorphic Gauss-Manin connection up to arbitrarily high order,
         and an algorithm of Gerard and Levelt (See R. Gerard, A.H.M. Levelt,
         Ann. Inst. Fourier, Grenoble 23,1 (1973), pp. 157-195) to transform
         it to a simple pole.
DISPLAY: The procedure displays more comments for higher printlevel.
EXAMPLE: example monodromy; shows an example.
"
{
  int opt;
  if(size(#)>0)
  {
    if(typeof(#[1])=="int")
    {
      opt=#[1];
    }
    else
    {
      print("\\second parameter no int");
      return();
    }
  }

  dbprint(printlevel-voice+2,"//basering="+string(basering));

  int i;
  for(i=nvars(basering);i>=1;i--)
  {
    if(1<var(i))
    {
      i=-1;
    }
  }

  if(i<0)
```

```
  {
    print("//no series ring (local ordering)");

    matrix M[1][0];
    return(M);
  }
  else
  {
    dbprint(printlevel-voice+2,"//f="+string(f));

    dbprint(printlevel-voice+2,"//computing milnor number mu of f...");
    int t=timer;
    int mu=milnor(f);
    dbprint(printlevel-voice+2,"//...mu computed ["+string(timer-t)+" secs, "
      +string((memory(1)+1023)/1024)+" K]");

    dbprint(printlevel-voice+2,"//mu="+string(mu));

    if(mu<=0)
    {
      if(mu==0)
      {
        print("//no singularity");
      }
      else
      {
        print("//non isolated singularity");
      }

      matrix M[1][0];
      return(M);
    }
    else
    {
      dbprint(printlevel-voice+2,"//computing weight vector w...");
      intvec w=qhweight(f);
      dbprint(printlevel-voice+2,"//...w computed");

      dbprint(printlevel-voice+2,"//w="+string(w));

      if(w==0)
      {
        dbprint(printlevel-voice+2,
          "//f not quasihomogeneous with respect to given coordinates");
        return(nonqhmonodromy(f,mu,opt));
      }
      else
      {
        dbprint(printlevel-voice+2,
          "//f quasihomogeneous with respect to given coordinates");
        return(qhmonodromy(f,w));
      }
    }
  }
}
example
{ "EXAMPLE:"; echo=2;
```

```
  ring R=0,(x,y),ds;
  poly f=x2y2+x6+y6;
  matrix M=monodromy(f);
  print(M);
}
///////////////////////////////////////////////////////////////////////////////

proc H''basis(poly f)
"USAGE:   H''basis(f); f poly
ASSUME:  The polynomial f in a series ring (local ordering) defines
         an isolated hypersurface singularity.
RETURN:  The procedure returns a list of representatives of a C{f}-basis of the
         Brieskorn lattice H''=Omega^(n+1)/df^dOmega^(n-1).
THEORY:  H'' is a free C{f}-module of rank milnor(f).
DISPLAY: The procedure displays more comments for higher printlevel.
EXAMPLE: example H''basis; shows an example.
"
{
  pcvinit();

  dbprint(printlevel-voice+2,"//basering="+string(basering));

  int i;
  for(i=nvars(basering);i>=1;i--)
  {
    if(1<var(i))
    {
      i=-1;
    }
  }

  if(i<0)
  {
    print("//no series ring (local ordering)");

    return(list());
  }
  else
  {
    dbprint(printlevel-voice+2,"//f="+string(f));

    dbprint(printlevel-voice+2,"//computing milnor number mu of f...");
    int t=timer;
    int mu=milnor(f);
    dbprint(printlevel-voice+2,"//...mu computed ["+string(timer-t)+" secs, "
      +string((memory(1)+1023)/1024)+" K]");

    dbprint(printlevel-voice+2,"//mu="+string(mu));

    if(mu<=0)
    {
      if(mu==0)
      {
        print("//no singularity");
      }
      else
      {
```

```
        print("//non isolated singularity");
      }

      return(list());
    }
    else
    {
      dbprint(printlevel-voice+2,"//computing kappa, xi and u with "+
        "u*f^kappa=(matrix(jacob(f))*xi)[1,1]...");
      list jl=jacoblift(f);
      def kappa,xi,u=jl[1..3];
      dbprint(printlevel-voice+2,"//...kappa, xi and u computed");
      dbprint(printlevel-voice+2,"//kappa="+string(kappa));
      if(kappa==1)
      {
        dbprint(printlevel-voice+2,
          "//f quasihomogenous with respect to suitable coordinates");
      }
      else
      {
        dbprint(printlevel-voice+2,
          "//f not quasihomogenous for any choice of coordinates");
      }
      dbprint(printlevel-voice+2,"//xi=");
      dbprint(printlevel-voice+2,xi);
      dbprint(printlevel-voice+2,"//u="+string(u));

      int K,N,prevN;
      list e,P1,P2,Pe,V1,V2,Ve,Vnablae;

      dbprint(printlevel-voice+2,"//increasing K and N...");
      K,N,P1,P2,Pe,V1,V2,Ve=incK(f,mu,K,1,N,e,P1,P2,Pe,V1,V2,Ve);
      dbprint(printlevel-voice+2,"//...K and N increased");

      dbprint(printlevel-voice+2,
        "//computing C{f}-basis e of Brieskorn lattice "
        +"H''=Omega^(n+1)/df^dOmega^(n-1)...");
      t=timer;
      e=pcvcv2p(quotV(V1+V2,N),0,N);
      dbprint(printlevel-voice+2,"//...e computed ["+string(timer-t)+" secs, "
        +string((memory(1)+1023)/1024)+" K]");

      dbprint(printlevel-voice+2,"//e=");
      dbprint(printlevel-voice+2,e);

      return(e);
    }
  }
}
example
{ "EXAMPLE:"; echo=2;
  ring R=0,(x,y),ds;
  poly f=x2y2+x6+y6;
  H''basis(f);
}
///////////////////////////////////////////////////////////////////////////////
```

4.4. **The** Singular **Library** `jordan.lib`**.** The Singular library `jordan.lib` provides procedures to compute the Jordan normal form of constant square matrices with eigenvalues in the coefficient field.

```
///////////////////////////////////////////////////////////////////////////////

version="$Id: jordan.lib,v 1.13 1999/06/22 15:39:03 mschulze Exp $";
info="
LIBRARY: jordan.lib  PROCEDURES TO COMPUTE THE JORDAN NORMAL FORM
AUTHOR:  Mathias Schulze, email: mschulze@mathematik.uni-kl.de

 jordan(M[,opt]);  eigenvalues, Jordan block sizes, Jordan transformation of M
 jordanmatrix(l);  Jordan matrix with eigenvalues l[1], Jordan block sizes l[2]
 jordanform(M);    Jordan normal form of constant square matrix M
 invmat(M);        inverse matrix of invertible constant matrix M
";

LIB "ring.lib";
///////////////////////////////////////////////////////////////////////////////

static proc countblocks(matrix M)
{
  int b,r,r0;

  int i=1;
  while(i<=nrows(M))
  {
    b++;
    r=nrows(M[i]);
    r0=r;

    dbprint(printlevel-voice+2,"//searching for block "+string(b)+"...");
    while(i<r0&&i<nrows(M))
    {
      i++;
      if(i<=nrows(M))
      {
        r=nrows(M[i]);
        if(r>r0)
        {
          r0=r;
        }
      }
    }
    dbprint(printlevel-voice+2,"//...block "+string(b)+" found");

    i++;
  }

  return(b);
}
///////////////////////////////////////////////////////////////////////////////

static proc getblock(matrix M,intvec v)
{
  matrix M0[size(v)][size(v)]=M[v,v];
  return(M0);
}
```

```
///////////////////////////////////////////////////////////////////////////////

proc jordan(matrix M,list #)
"USAGE:   jordan(M[,opt]); M constant square matrix, opt integer
ASSUME:  The eigenvalues of M are in the coefficient field.
RETURN:  The procedure returns a list jd with 3 entries of type
         ideal, list of intvecs, matrix with
         jd[1] eigenvalues of M,
         jd[2][i][j] size of j-th Jordan block with eigenvalue jd[1][i], and
         jd[3]^(-1)*M*jd[3] in Jordan normal form.
         Depending on opt, only certain entries of jd are computed.
         If opt=-1, jd[1] is computed,
         if opt= 0, jd[1] and jd[2] are computed,
         if opt= 1, jd[1], jd[2], and jd[3] are computed, and,
         if opt= 2, jd[1] and jd[3] are computed.
         By default, opt=0.
NOTE:    A non constant polynomial matrix M is replaced by its constant part.
DISPLAY: The procedure displays comments if printlevel>=1.
EXAMPLE: example jordan; shows an example.
"
{
  int n=nrows(M);
  if(n==0)
  {
    print("//empty matrix");
    return(list());
  }
  if(n!=ncols(M))
  {
    print("//no square matrix");
    return(list());
  }

  M=jet(M,0);

  dbprint(printlevel-voice+2,"//counting blocks of matrix...");
  int i=countblocks(M);
  dbprint(printlevel-voice+2,"//...blocks of matrix counted");
  if(i==1)
  {
    dbprint(printlevel-voice+2,"//matrix has 1 block");
  }
  else
  {
    dbprint(printlevel-voice+2,"//matrix has "+string(i)+" blocks");
  }

  dbprint(printlevel-voice+2,"//counting blocks of transposed matrix...");
  int j=countblocks(transpose(M));
  dbprint(printlevel-voice+2,"//...blocks of transposed matrix counted");
  if(j==1)
  {
    dbprint(printlevel-voice+2,"//transposed matrix has 1 block");
  }
  else
  {
    dbprint(printlevel-voice+2,"//transposed matrix has "+string(j)+" blocks");
```

```
}

if(i<j)
{
  dbprint(printlevel-voice+2,"//transposing matrix...");
  M=transpose(M);
  dbprint(printlevel-voice+2,"//...matrix transposed");
}

list fd;
matrix M0;
poly cp;
ideal eM,eM0;
intvec mM,mM0;
intvec u;
int b,r,r0;

i=1;
while(i<=nrows(M))
{
  b++;
  u=i;
  r=nrows(M[i]);
  r0=r;

  dbprint(printlevel-voice+2,"//searching for block "+string(b)+"...");
  while(i<r0&&i<nrows(M))
  {
    i++;
    if(i<=nrows(M))
    {
      u=u,i;
      r=nrows(M[i]);
      if(r>r0)
      {
        r0=r;
      }
    }
  }
  dbprint(printlevel-voice+2,"//...block "+string(b)+" found");

  if(size(u)==1)
  {
    dbprint(printlevel-voice+2,"//1x1-block:");
    dbprint(printlevel-voice+2,M[u[1]][u[1]]);

    if(mM[1]==0)
    {
      eM=M[u[1]][u[1]];
      mM=1;
    }
    else
    {
      eM=eM,ideal(M[u[1]][u[1]]);
      mM=mM,1;
    }
  }
```

```
  else
  {
    dbprint(printlevel-voice+2,
      "//"+string(size(u))+"x"+string(size(u))+"-block:");
    M0=getblock(M,u);
    dbprint(printlevel-voice+2,M0);

    dbprint(printlevel-voice+2,"//characteristic polynomial:");
    cp=det(module(M0-var(1)*freemodule(size(u))));
    dbprint(printlevel-voice+2,cp);

    dbprint(printlevel-voice+2,"//factorizing characteristic polynomial...");
    fd=factorize(cp,2);
    dbprint(printlevel-voice+2,"//...characteristic polynomial factorized");

    dbprint(printlevel-voice+2,"//computing eigenvalues...");
    eM0,mM0=fd[1..2];
    if(1<var(1))
    {
      for(j=ncols(eM0);j>=1;j--)
      {
        if(deg(eM0[j])>1)
        {
          print("//eigenvalues not in the coefficient field");
          return(list());
        }
        eM0[j]=-(eM0[j][2]/var(1))/eM0[j][1];
      }
    }
    else
    {
      for(j=ncols(eM0);j>=1;j--)
      {
        if(deg(eM0[j])>1)
        {
          print("//eigenvalues not in the coefficient field");
          return(list());
        }
        eM0[j]=-eM0[j][1]/(eM0[j][2]/var(1));
      }
    }
    dbprint(printlevel-voice+2,"//...eigenvalues computed");

    if(mM[1]==0)
    {
      eM=eM0;
      mM=mM0;
    }
    else
    {
      eM=eM,eM0;
      mM=mM,mM0;
    }
  }

  i++;
}
```

```
dbprint(printlevel-voice+2,"//sorting eigenvalues...");
poly e;
int m;
for(i=ncols(eM);i>=2;i--)
{
  for(j=i-1;j>=1;j--)
  {
   if(eM[i]<eM[j])
    {
      e=eM[i];
      eM[i]=eM[j];
      eM[j]=e;
      m=mM[i];
      mM[i]=mM[j];
      mM[j]=m;
    }
  }
}
dbprint(printlevel-voice+2,"//...eigenvalues sorted");

dbprint(printlevel-voice+2,"//removing multiple eigenvalues...");
i=1;
j=2;
while(j<=ncols(eM))
{
  if(eM[i]==eM[j])
  {
    mM[i]=mM[i]+mM[j];
  }
  else
  {
    i++;
    eM[i]=eM[j];
    mM[i]=mM[j];
  }
  j++;
}
eM=eM[1..i];
mM=mM[1..i];
dbprint(printlevel-voice+2,"//...multiple eigenvalues removed");

dbprint(printlevel-voice+2,"//eigenvalues:");
dbprint(printlevel-voice+2,eM);
dbprint(printlevel-voice+2,"//multiplicities:");
dbprint(printlevel-voice+2,mM);

int opt=0;
if(size(#)>0)
{
  if(typeof(#[1])=="int")
  {
    opt=#[1];
  }
}
if(opt<0)
{
```

```
  return(list(eM));
}
int k,l;
matrix I=freemodule(n);
matrix Mi,Ni;
module sNi;
list K;
if(opt>=1)
{
  module V,K1,K2;
  matrix v[n][1];
}
if(opt<=1)
{
  list bM;
  intvec bMi;
}

for(i=ncols(eM);i>=1;i--)
{
  Mi=M-eM[i]*I;

  dbprint(printlevel-voice+2,
    "//computing kernels of powers of matrix minus eigenvalue "
    +string(eM[i]));
  K=list(module());
  for(Ni,sNi=Mi,0;size(sNi)<mM[i];Ni=Ni*Mi)
  {
    sNi=syz(Ni);
    K=K+list(sNi);
  }
  dbprint(printlevel-voice+2,"//...kernels computed");

  if(opt<=1)
  {
    dbprint(printlevel-voice+2,
      "//computing Jordan block sizes for eigenvalue "
      +string(eM[i])+"...");
    bMi=0;
    bMi[size(K[2])]=0;
    for(j=size(K);j>=2;j--)
    {
      for(k=size(bMi);k>size(bMi)+size(K[j-1])-size(K[j]);k--)
      {
        bMi[k]=bMi[k]+1;
      }
    }
    bM=list(bMi)+bM;
    dbprint(printlevel-voice+2,"//...Jordan block sizes computed");
  }

  if(opt>=1)
  {
    dbprint(printlevel-voice+2,
      "//computing Jordan basis vectors for eigenvalue "
      +string(eM[i])+"...");
    if(size(K)>1)
```

```
      {
        for(j,K1=2,0;j<=size(K)-1;j++)
        {
          K2=K[j];
          K[j]=interred(reduce(K[j],std(K1+module(Mi*K[j+1]))));
          K1=K2;
        }
        K[j]=interred(reduce(K[j],std(K1)));
      }
      for(j=size(K);j>=2;j--)
      {
        for(k=size(K[j]);k>=1;k--)
        {
          v=K[j][k];
          for(l=j;l>=1;l--)
          {
            V=module(v)+V;
            v=Mi*v;
          }
        }
      }
      dbprint(printlevel-voice+2,"//...Jordan basis vectors computed");
    }
  }

  list jd=eM;
  if(opt<=1)
  {
    jd[2]=bM;
  }
  if(opt>=1)
  {
    jd[3]=V;
  }
  return(jd);
}
example
{ "EXAMPLE:"; echo=2;
  ring R=0,x,dp;
  matrix M[3][3]=3,2,1,0,2,1,0,0,3;
  print(M);
  jordan(M);
}
///////////////////////////////////////////////////////////////////////////////

proc jordanmatrix(list jd)
"USAGE:   jordanmatrix(jd); jd list of ideal and list of intvecs
RETURN:  The procedure returns the Jordan matrix J with eigenvalues jd[1] and
         size jd[2][i][j] of j-th Jordan block with eigenvalue jd[1][i].
DISPLAY: The procedure displays comments if printlevel>=1.
EXAMPLE: example jordanmatrix; shows an example.
"
{
  if(size(jd)<2)
  {
    print("//not enough entries in argument list");
    matrix J[1][0];
```

```
  return(J);
}
def eJ,bJ=jd[1..2];
if(typeof(eJ)!="ideal")
{
  print("//first entry in argument list not an ideal");
  matrix J[1][0];
  return(J);
}
if(typeof(bJ)!="list")
{
  print("//second entry in argument list not a list");
  matrix J[1][0];
  return(J);
}
if(size(eJ)<size(bJ))
{
  int s=size(eJ);
}
else
{
  int s=size(bJ);
}

int i,j,k,n;
for(i=s;i>=1;i--)
{
  if(typeof(bJ[i])!="intvec")
  {
    print("//second entry in argument list not a list of intvecs");
    matrix J[1][0];
    return(J);
  }
  else
  {
    for(j=size(bJ[i]);j>=1;j--)
    {
      k=bJ[i][j];
      if(k>0)
      {
        n=n+k;
      }
    }
  }
}

int l;
matrix J[n][n];
for(i,l=1,1;i<=s;i++)
{
  for(j=1;j<=size(bJ[i]);j++)
  {
    k=bJ[i][j];
    if(k>0)
    {
      while(k>=2)
      {
```

```
          J[l,l]=eJ[i];
          J[l,l+1]=1;
          k,l=k-1,l+1;
        }
        J[l,l]=eJ[i];
        l++;
      }
    }
  }

  return(J);
}
example
{ "EXAMPLE:"; echo=2;
  ring R=0,x,dp;
  list l;
  l[1]=ideal(2,3);
  l[2]=list(intvec(1),intvec(2));
  print(jordanmatrix(l));
}
///////////////////////////////////////////////////////////////////////////////

proc jordanform(matrix M)
"USAGE:   jordanform(M); M constant square matrix
ASSUME:  The eigenvalues of M are in the coefficient field.
RETURN:  The procedure returns the Jordan normal form of M.
NOTE:    A non constant polynomial matrix M is replaced by its constant part.
DISPLAY: The procedure displays more comments for higher printlevel.
EXAMPLE: example jordanform; shows an example.
"
{
  return(jordanmatrix(jordan(M)));
}
example
{ "EXAMPLE:"; echo=2;
  ring R=0,x,dp;
  matrix M[3][3]=3,2,1,0,2,1,0,0,3;
  print(M);
  print(jordanform(M));
}
///////////////////////////////////////////////////////////////////////////////

proc invmat(matrix M)
"USAGE:   invmat(M); M constant square matrix
ASSUME:  M is invertible.
RETURN:  The procedure returns the inverse matrix of M.
NOTE:    A non constant polynomial matrix M is replaced by its constant part.
EXAMPLE: example invmat; shows an example.
"
{
  if(nrows(M)==ncols(M))
  {
    matrix invM=lift(jet(M,0),freemodule(nrows(M)));
  }
  else
  {
    print("//no square matrix");
```

```
    matrix[1][0]=invM;
  }
  return(invM);
}
example
{ "EXAMPLE:"; echo=2;
  ring R=0,x,dp;
  matrix M[3][3]=3,2,1,0,2,1,0,0,3;
  print(M);
  print(invmat(M));
}
///////////////////////////////////////////////////////////////////////////////
```

4.5. **The SINGULAR Extension pcv.** The SINGULAR extension pcv provides procedures to convert between **p**olynomials and **c**oefficient **v**ectors and can be linked as a dynamic module. Using pcv and the arithmetic for modules, SINGULAR can do linear algebra on finite dimensional algebras.

4.5.1. pcv.h.

```
/****************************************
*  Computer Algebra System SINGULAR     *
****************************************/
/* $Id: pcv.h,v 1.13 1999/06/11 17:13:33 mschulze Exp $ */
/*
* ABSTRACT: conversion between polys and coef vectors
*/

#ifndef PCV_H
#define PCV_H

lists pcvLAddL(lists l1,lists l2);
lists pcvPMulL(poly p,lists l1);
BOOLEAN pcvLAddL(leftv res,leftv h);
BOOLEAN pcvPMulL(leftv res,leftv h);
int pcvDeg(poly p);
int pcvMinDeg(poly p);
int pcvMinDeg(matrix m);
BOOLEAN pcvMinDeg(leftv res,leftv h);
void pcvInit(int d);
void pcvClean();
int pcvM2N(poly m);
poly pcvN2M(int n);
poly pcvP2CV(poly p,int d0,int d1);
poly pcvCV2P(poly cv,int d0,int d1);
lists pcvP2CV(lists pl,int d0,int d1);
ideal pcvP2CV(ideal p,int d0,int d1);
lists pcvCV2P(lists cvl,int d0,int d1);
ideal pcvCV2P(ideal cv,int d0,int d1);
BOOLEAN pcvP2CV(leftv res,leftv h);
BOOLEAN pcvCV2P(leftv res,leftv h);
int pcvDim(int d0,int d1);
BOOLEAN pcvDim(leftv res,leftv h);
int pcvBasis(lists b,int i,poly m,int d,int n);
lists pcvBasis(int d0,int d1);
BOOLEAN pcvBasis(leftv res,leftv h);

#endif
```

4.5.2. pcv.cc.

```
/****************************************
*  Computer Algebra System SINGULAR      *
****************************************/
/* $Id: pcv.cc,v 1.25 1999/06/11 17:13:40 mschulze Exp $ */
/*
* ABSTRACT: conversion between polys and coef vectors
*/

#include "mod2.h"
```

```
#ifdef HAVE_PCV
#if !defined(HAVE_DYNAMIC_LOADING) || defined(BUILD_MODULE)

#include "tok.h"
#include "ipid.h"
#include "numbers.h"
#include "polys.h"
#include "ideals.h"
#include "lists.h"
#include "matpol.h"
#include "febase.h"
#include "pcv.h"

static int pcvMaxDegree;
static int pcvTableSize;
static int pcvIndexSize;
static unsigned* pcvTable=NULL;
static unsigned** pcvIndex=NULL;

lists pcvLAddL(lists l1,lists l2)
{
  lists l0=(lists)Alloc(sizeof(slists));
  int i=l1->nr;
  if(l1->nr<l2->nr) i=l2->nr;
  l0->Init(i+1);
  for(;i>=0;i--)
  {
    if(i<=l1->nr&&(l1->m[i].rtyp==POLY_CMD||l1->m[i].rtyp==VECTOR_CMD))
    {
      l0->m[i].rtyp=l1->m[i].rtyp;
      l0->m[i].data=pCopy((poly)l1->m[i].data);
      if(i<=l2->nr&&l2->m[i].rtyp==l1->m[i].rtyp)
        l0->m[i].data=pAdd(l0->m[i].data,pCopy((poly)l2->m[i].data));
    }
    else
    if(i<=l2->nr&&(l2->m[i].rtyp==POLY_CMD||l2->m[i].rtyp==VECTOR_CMD))
    {
      l0->m[i].rtyp=l2->m[i].rtyp;
      l0->m[i].data=pCopy((poly)l2->m[i].data);
    }
  }
  return(l0);
}

lists pcvPMulL(poly p,lists l1)
{
  lists l0=(lists)Alloc(sizeof(slists));
  l0->Init(l1->nr+1);
  for(int i=l1->nr;i>=0;i--)
  {
    if(l1->m[i].rtyp==POLY_CMD)
    {
      l0->m[i].rtyp=POLY_CMD;
      l0->m[i].data=pMult(pCopy(p),pCopy((poly)l1->m[i].data));
    }
  }
  return(l0);
```

```
}

BOOLEAN pcvLAddL(leftv res,leftv h)
{
  if(h&&h->Typ()==LIST_CMD)
  {
    lists l1=(lists)h->Data();
    h=h->next;
    if(h&&h->Typ()==LIST_CMD)
    {
      lists l2=(lists)h->Data();
      res->rtyp=LIST_CMD;
      res->data=(void*)pcvLAddL(l1,l2);
      return FALSE;
    }
  }
  WerrorS("<list>,<list> expected");
  return TRUE;
}

BOOLEAN pcvPMulL(leftv res,leftv h)
{
  if(h&&h->Typ()==POLY_CMD)
  {
    poly p=(poly)h->Data();
    h=h->next;
    if(h&&h->Typ()==LIST_CMD)
    {
      lists l=(lists)h->Data();
      res->rtyp=LIST_CMD;
      res->data=(void*)pcvPMulL(p,l);
      return FALSE;
    }
  }
  WerrorS("<poly>,<list> expected");
  return TRUE;
}

int pcvDeg(poly p)
{
  int d=0;
  for(int i=pVariables;i>=1;i--) d+=pGetExp(p,i);
  return d;
}

int pcvMinDeg(poly p)
{
  if(!p) return -1;
  int md=pcvDeg(p);
  pIter(p);
  while(p)
  {
    int d=pcvDeg(p);
    if(d<md) md=d;
    pIter(p);
  }
  return md;
```

```
}

int pcvMinDeg(matrix m)
{
  int i,j,d;
  int md=-1;
  for(i=1;i<=MATROWS(m);i++)
  {
    for(j=1;j<=MATCOLS(m);j++)
    {
      d=pcvMinDeg(MATELEM(m,i,j));
      if((d>=0&&md>d)||md==-1) md=d;
    }
  }
  return(md);
}

BOOLEAN pcvMinDeg(leftv res,leftv h)
{
  if(h)
  {
    if(h->Typ()==POLY_CMD)
    {
      res->rtyp=INT_CMD;
      res->data=(void*)pcvMinDeg((poly)h->Data());
      return FALSE;
    }
    else
    if(h->Typ()==MATRIX_CMD)
    {
      res->rtyp=INT_CMD;
      res->data=(void*)pcvMinDeg((matrix)h->Data());
      return FALSE;
    }
  }
  WerrorS("<poly> expected");
  return TRUE;
}

void pcvInit(int d)
{
  if(d<0) d=1;
  pcvMaxDegree=d+1;
  pcvTableSize=pVariables*pcvMaxDegree*sizeof(unsigned);
  pcvTable=(unsigned*)Alloc0(pcvTableSize);
  pcvIndexSize=pVariables*sizeof(unsigned*);
  pcvIndex=(unsigned**)Alloc(pcvIndexSize);
  for(int i=0;i<pVariables;i++)
    pcvIndex[i]=pcvTable+i*pcvMaxDegree;
  for(int i=0;i<pcvMaxDegree;i++)
    pcvIndex[0][i]=i;
  unsigned k,l;
  for(int i=1;i<pVariables;i++)
  {
    k=0;
    for(int j=0;j<pcvMaxDegree;j++)
    {
```

```
      l=pcvIndex[i-1][j];
      if(l>unsigned(~0)-k)
      {
        j=pcvMaxDegree;
        i=pVariables;
        WerrorS("unsigned overflow");
      }
      else pcvIndex[i][j]=k+=l;
    }
  }
}

void pcvClean()
{
  if(pcvTable)
  {
    Free(pcvTable,pcvTableSize);
    pcvTable=NULL;
  }
  if(pcvIndex)
  {
    Free(pcvIndex,pcvIndexSize);
    pcvIndex=NULL;
  }
}

int pcvM2N(poly m)
{
  unsigned n=0,dn,d=0;
  for(int i=0;i<pVariables;i++)
  {
    d+=pGetExp(m,i+1);
    dn=pcvIndex[i][d];
    if(dn>MAX_COMPONENT-n)
    {
      i=pVariables;
      WerrorS("component overflow");
    }
    else n+=dn;
  }
  return n+1;
}

poly pcvN2M(int n)
{
  n--;
  poly m=pOne();
  int i,j,k;
  for(i=pVariables-1;i>=0;i--)
  {
    k=j;
    for(j=0;j<pcvMaxDegree&&pcvIndex[i][j]<=n;j++);
    j--;
    n-=pcvIndex[i][j];
    if(i<pVariables-1) pSetExp(m,i+2,k-j);
  }
  if(n==0)
```

```
  {
    pSetExp(m,1,j);
    pSetm(m);
    return m;
  }
  else
  {
    pDelete1(&m);
    return NULL;
  }
}

poly pcvP2CV(poly p,int d0,int d1)
{
  poly cv=NULL;
  while(p)
  {
    int d=pcvDeg(p);
    if(d0<=d&&d<d1)
    {
      poly c=pOne();
      pSetComp(c,pcvM2N(p));
      pSetCoeff(c,nCopy(pGetCoeff(p)));
      cv=pAdd(cv,c);
    }
    pIter(p);
  }
  return cv;
}

poly pcvCV2P(poly cv,int d0,int d1)
{
  poly p=NULL;
  while(cv)
  {
    poly m=pcvN2M(pGetComp(cv));
    if(m)
    {
      int d=pcvDeg(m);
      if(d0<=d&&d<d1)
      {
        pSetCoeff(m,nCopy(pGetCoeff(cv)));
        p=pAdd(p,m);
      }
    }
    pIter(cv);
  }
  return p;
}

lists pcvP2CV(lists pl,int d0,int d1)
{
  lists cvl=(lists)Alloc(sizeof(slists));
  cvl->Init(pl->nr+1);
  pcvInit(d1);
  for(int i=pl->nr;i>=0;i--)
  {
```

```
    if(pl->m[i].rtyp==POLY_CMD)
    {
      cvl->m[i].rtyp=VECTOR_CMD;
      cvl->m[i].data=pcvP2CV((poly)pl->m[i].data,d0,d1);
    }
  }
  pcvClean();
  return cvl;
}

lists pcvCV2P(lists cvl,int d0,int d1)
{
  lists pl=(lists)Alloc(sizeof(slists));
  pl->Init(cvl->nr+1);
  pcvInit(d1);
  for(int i=cvl->nr;i>=0;i--)
  {
    if(cvl->m[i].rtyp==VECTOR_CMD)
    {
      pl->m[i].rtyp=POLY_CMD;
      pl->m[i].data=pcvCV2P((poly)cvl->m[i].data,d0,d1);
    }
  }
  pcvClean();
  return pl;
}

BOOLEAN pcvP2CV(leftv res,leftv h)
{
  if(currRingHdl)
  {
    if(h&&h->Typ()==LIST_CMD)
    {
      lists p=(lists)h->Data();
      h=h->next;
      if(h&&h->Typ()==INT_CMD)
      {
        int d0=(int)h->Data();
        h=h->next;
        if(h&&h->Typ()==INT_CMD)
        {
          int d1=(int)h->Data();
          res->rtyp=LIST_CMD;
          res->data=(void*)pcvP2CV(p,d0,d1);
          return FALSE;
        }
      }
    }
    WerrorS("<list>,<int>,<int> expected");
    return TRUE;
  }
  WerrorS("no ring active");
  return TRUE;
}

BOOLEAN pcvCV2P(leftv res,leftv h)
{
```

```
  if(currRingHdl)
  {
    if(h&&h->Typ()==LIST_CMD)
    {
      lists pl=(lists)h->Data();
      h=h->next;
      if(h&&h->Typ()==INT_CMD)
      {
        int d0=(int)h->Data();
        h=h->next;
        if(h&&h->Typ()==INT_CMD)
        {
          int d1=(int)h->Data();
          res->rtyp=LIST_CMD;
          res->data=(void*)pcvCV2P(pl,d0,d1);
          return FALSE;
        }
      }
    }
    WerrorS("<list>,<int>,<int> expected");
    return TRUE;
  }
  WerrorS("no ring active");
  return TRUE;
}

int pcvDim(int d0,int d1)
{
  if(d0<0) d0=0;
  if(d1<0) d1=0;
  pcvInit(d1);
  int d=pcvIndex[pVariables-1][d1]-pcvIndex[pVariables-1][d0];
  pcvClean();
  return d;
}

BOOLEAN pcvDim(leftv res,leftv h)
{
  if(currRingHdl)
  {
    if(h&&h->Typ()==INT_CMD)
    {
      int d0=(int)h->Data();
      h=h->next;
      if(h&&h->Typ()==INT_CMD)
      {
        int d1=(int)h->Data();
        res->rtyp=INT_CMD;
        res->data=(void*)pcvDim(d0,d1);
        return FALSE;
      }
    }
    WerrorS("<int>,<int> expected");
    return TRUE;
  }
  WerrorS("no ring active");
  return TRUE;
```

```
}

int pcvBasis(lists b,int i,poly m,int d,int n)
{
  if(n<pVariables)
  {
    for(int k=0,l=d;k<=l;k++,d--)
    {
      pSetExp(m,n,k);
      i=pcvBasis(b,i,m,d,n+1);
    }
  }
  else
  {
    pSetExp(m,n,d);
    pSetm(m);
    b->m[i].rtyp=POLY_CMD;
    b->m[i++].data=pCopy(m);
  }
  return i;
}

lists pcvBasis(int d0,int d1)
{
  if(d0<0) d0=0;
  if(d1<0) d1=0;
  lists b=(lists)Alloc(sizeof(slists));
  b->Init(pcvDim(d0,d1));
  poly m=pOne();
  for(int d=d0,i=0;d<d1;d++)
    i=pcvBasis(b,i,m,d,1);
  pDelete1(&m);
  return b;
}

BOOLEAN pcvBasis(leftv res,leftv h)
{
  if(currRingHdl)
  {
    if(h&&h->Typ()==INT_CMD)
    {
      int d0=(int)h->Data();
      h=h->next;
      if(h&&h->Typ()==INT_CMD)
      {
        int d1=(int)h->Data();
        res->rtyp=LIST_CMD;
        res->data=(void*)pcvBasis(d0,d1);
        return FALSE;
      }
    }
    WerrorS("<int>,<int> expected");
    return TRUE;
  }
  WerrorS("no ring active");
  return TRUE;
}
```

```
#endif /* !defined(HAVE_DYNAMIC_LOADING) || defined(BUILD_MODULE) */
#endif /* HAVE_PCV */
```

References

[B+87] A. Borel et al. (eds.), *Algebraic D-Modules*, 2. ed., Perspectives in Mathematics, vol. 2, Academic Press, Boston, 1987.

[Bre97] G. E. Bredon, *Sheaf Theory*, 2. ed., Graduate Texts in Mathematics, vol. 170, Springer, Heidelberg, 1997.

[Bri70] E. Brieskorn, *Die Monodromie der isolierten Singularitäten von Hyperflächen*, manuscripta math. **2** (1970), 103–161.

[CL55] E. A. Coddington and N. Levinson, *Theory of Ordinary Differential Equations*, McGraw-Hill, New York, 1955.

[Del70] P. Deligne, *Équations Différentielles à Points Singulier Réguliers*, Lecture Notes in Mathematics, vol. 163, Springer, Berlin, 1970.

[Eis96] D. Eisenbud, *Commutative Algebra with a View toward Algebraic Geometry*, Graduate Texts in Mathematics, vol. 150, Springer, Heidelberg, 1996.

[GL73] R. Gérard and A. H. M. Levelt, *Invariants Mésurant l'Irrégularité en un Point Singulier des Systèmes d'Équations Différentielles Linéaires*, Ann. Inst. Fourier, Grenoble **23** (1973), no. 1, 157–195.

[God64] R. Godement, *Topologie Algébrique et Théorie des Faisceaux*, 3. ed., Actualités Scientifiques et Industrielles, vol. 1252, Hermann, Paris, 1964.

[GPS98] G.-M. Greuel, G. Pfister, and H. Schönemann, *Singular version 1.2 User Manual*, Reports On Computer Algebra, no. 21, Centre for Computer Algebra, University of Kaiserslautern, June 1998, `http://www.mathematik.uni-kl.de/~zca/Singular`

[Gre71] G.-M. Greuel, *Zur Picard-Lefschetz-Monodromie isolierter Singularitäten von vollständigen Durchschnitten*, Diplomarbeit, Göttingen, 1971.

[Gre75] G.-M. Greuel, *Der Gauss-Manin-Zusammenhang isolierter Singularitäten von vollständigen Durchschnitten*, Math. Ann. **214** (1975), 235–266.

[Mal74] B. Malgrange, *Sur les points singuliers des équations différentielles*, L'enseignement mathématique **20** (1974), 147–176.

[Mil68] J. Milnor, *Singular Points on Complex Hypersurfaces*, Annals of Mathematics Studies, vol. 61, Princeton University Press, Princeton, NJ, 1968.

[Nac90] P. F. M. Nacken, *A Computer Program for the Computation of the Monodromy of an Isolated Singularity*, Master's thesis, Department of Mathematics, Catholic Universtity, Toernooiveld, 6525 ED Nijmegen, The Netherlands, March 1990.

[Seb70] M. Sebastiani, *Preuve d'une Conjecture de Brieskorn*, manuscripta math. **2** (1970), 301–308.

Mathias Schulze, Jakob-Locher-Strasse 16, 66482 Zweibrücken
E-mail address: `mschulze@mathematik.uni-kl.de`