ISSN 1349-127X (Online)
ISSB 1349-1261 (Paper)
http://www.math.kobe-u.ac.jp/raj

Risa/Asir Journal

Volume 4 (2012)

Risa/Asir Journal will be a step to "Journal of Free Mathematical Software". We welcome not only articles related to Computer algebra system Risa/Asir, but also any articles on free mathematical software.

Please visit `http://www.math.kobe-u.ac.jp/raj`. All articles and attached software systems will be reviewed.

平面曲線に対する twisted logarithmic cohomology 群の計算アルゴリズムとその超幾何積分への応用

西谷圭祐, 神戸大学理学研究科

平成 24 年 2 月 29 日

1 はじめに

$R = \mathbb{C}[x_1, \cdots, x_n]$ を n 変数多項式環とする.

$(\Omega_R^\bullet, d)$ を R 係数（または正則）微分形式のなす複体とする. ただし d は外微分を表す. 0 でない多項式 $f \in R$ に対し, R_f で R を f で局所化した環を表す. すなわち

$$R_f = \{\frac{g}{f^m} \mid g \in R,\ m \in \mathbb{N}\}.$$

$(\Omega_f^\bullet, d) := (R_f \otimes \Omega_R^\bullet, d)$ を R_f 係数の有理微分形式のなす複体とする.

f を定数でない無平方多項式とする. 齋藤 [25] に従い, 有理 p 次微分形式 $\omega \in \Omega_f^p$ が条件 $f\omega \in \Omega_R^p$, $df \wedge \omega \in \Omega_R^{p+1}$ を満たすとき, 対数 p 次微分形式といい, 対数 p 次微分形式全体のなす R-加群を $\Omega_R^p(\log f)$ (または単に $\Omega^p(\log f)$) と表す. f の既約分解を $f = f_1 \cdots f_s$ とし, $\alpha_1, \cdots, \alpha_s \in \mathbb{C}$ をパラメータする. $\Phi = \prod_{i=1}^s f_i^{\alpha_i}$ とおく. このとき, $\nabla := d + d\log\Phi\wedge = d + \sum_{i=1}^s \alpha_i \frac{df_i}{f_i}\wedge$ に対して

$$0 \longrightarrow \Omega^0(\log f) \xrightarrow{\nabla} \Omega^1(\log f) \xrightarrow{\nabla} \cdots \xrightarrow{\nabla} \Omega^n(\log f) \longrightarrow 0$$

は複体をなす (d の場合の証明は [25, 1.3] で与えられた. ∇ の場合の証明は [2, p 70] を参照). この複体から定まるコホモロジー群を twisted logarithmic cohomology 群といい, $H^\bullet(\Omega^\bullet(\log f), \nabla)$ と書く.

Castro, 高山 [8] は平面曲線に対する logarithmic cohomology 群を計算するアルゴリズムを与えた. 彼らのアルゴリズムでは logarithmic cohomology 群の基底を求めるために $\mathrm{Der}(\log f)$ の自由基底を計算する必要があり, そのために多項式環上の射影加群の自由基底を求める Quillen-Suslin アルゴリズム [13] を用いていた. 本論文では, twisted logarithmic cohomology 群を計算する新しいアルゴリズムを与える. 我々のアルゴリズムには Quillen-Suslin アルゴリズムを必要としないため, Castro-高山アルゴリズムより高速なアル

ゴリズムとなっている. 我々のアルゴリズムでは左 D-加群の自由分解の間の鎖準同型の構成アルゴリズムを利用している. アルゴリズムは数式処理システム Macaulay2 [30] 上で実装した.

さらに, 本論文では twisted logarithmic cohomology 群の計算アルゴリズムの応用として, 多項式ベキの積分から定まるあるクラスの超幾何積分の満たす差分方程式系を求めるアルゴリズムと, 指数関数と多項式ベキの積分から定まるあるクラスの超幾何積分の満たす微分方程式系を求めるアルゴリズムを与える. 超幾何積分が満たす微分方程式, 差分方程式を求めるアルゴリズムには, 代表的なものとして Zeilberger アルゴリズム [1, 27, 28], Chyzak アルゴリズム [5, 6, 7], 積分アルゴリズム [16, 18, 20, 22] がある. Zeilberger アルゴリズムは creative-telescoping の方法を用いて, hyperexponential と呼ばれるクラスの関数の積分が満たす微分方程式系, 差分方程式系を計算するアルゴリズムである. Chyzak アルゴリズムは, Ore 代数上の Gröbner 基底と未定係数法により, ホロノミックな関数の積分が満たす微分方程式系, 差分方程式系を計算するアルゴリズムである. D-加群の積分アルゴリズムは大阿久 [18] によって与えられた. これの応用として, 微分作用素環における Gröbner 基底を用いて, ホロノミックな関数の満たす微分方程式系 (または微差分方程式系) から積分の満たす微分方程式系 (または差分方程式系) を得ることができる. 上記のアルゴリズムに対して, 今回新たに得られたアルゴリズムでは twisted logarithmic cohomology 群の基底を用いてあるクラスの超幾何積分の満たす微分方程式, 差分方程式を計算する. このアルゴリズムは積分アルゴリズムに基づいているが, 積分すべき D の左イデアルが, 本来の積分アルゴリズムの方法とは異なるものとなっている. アルゴリズムの実装は数式処理システム Risa/Asir [35] 上で行った. 我々は, 我々が実装したアルゴリズムと Maple に実装されている Chyzak アルゴリズム [29] との計算時間の比較を行った. その結果, 我々のアルゴリズムの方が速く計算できる場合があることが分かった.

2 自由分解の間の鎖準同型の構成アルゴリズム

2.1 Ore 代数の定義

まず Ore 代数の定義を復習しておく. Ore 代数は Ore [24] で導入された. Ore 代数は微分作用素環や差分作用素環を一般化したものである.

定義 2.1. (**Ore** 代数 [7])

1. R を整域とする. 非可換環 $R[\partial; \sigma, \delta]$ が以下の条件を満たすとき, *skew polynomial ring* という:

   i. $\partial a = \sigma(a)\partial + \delta(a), \quad a \in R,$

ii. σ, δ はそれぞれ R の自己準同型, σ-derivation である.

2. $R = K[x_1, \cdots, x_n]$ を標数 0 の体 K 上の多項式環とする. skew polynomial ring $O = R[\partial_1; \sigma_1, \delta_1]...[\partial_m; \sigma_m, \delta_m]$ が次の条件を満たすとき, O を *Ore* 代数という: σ_i, δ_i $(1 \leq i, j \leq m)$ は可換であり, 交換関係

$$\sigma_i(\partial_j) = \partial_j,\ \delta_i(\partial_j) = 0,\ j < i$$

が成り立つ.

例 **2.2.** $D = D_n = K\langle x_1, \cdots, x_n, \partial_{x_1}, \cdots, \partial_{x_n}\rangle$ n 変数微分作用素環とする. すなわち, D_n は $x_1, \cdots, x_n$ と $\partial_{x_1}, \cdots, \partial_{x_n}$ で生成される K 代数であり, 次の交換関係を満たす:

$$x_i x_j = x_j x_i,\quad \partial_{x_i}\partial_{x_j} = \partial_{x_j}\partial_{x_i},\quad \partial_{x_i} x_j = x_j \partial_{x_i} + \delta_{ij}\ (i, j = i, \cdots, n).$$

$\partial_i = \partial_{x_i}, \sigma_i = 1, \delta_i = \partial_i$ とすると, 微分作用素環は Ore 代数となる.

例 **2.3.** $E_s = K(\alpha_1, \cdots, \alpha_s)\langle E_{\alpha_1}, \cdots, E_{\alpha_s}\rangle$ を s 変数差分作用素環とする. すなわち次の交換関係を満たす:

$$E_{\alpha_i} E_{\alpha_j} = E_{\alpha_j} E_{\alpha_i},\ E_{\alpha_i} p(\alpha) = p(\alpha + e_i) E_{\alpha_i}\ (p(\alpha) \in K(\alpha_1, \cdots, \alpha_s)).$$

ただし, $i, j = 1, \cdots, s$. $\partial_i = E_{\alpha_i}, \sigma_i = E_{\alpha_i}, \delta_i = 0$ とすると, 差分作用素環は Ore 代数となる.

Ore 代数の左イデアルに対して, Gröbner 基底, Buchberger アルゴリズム, シヂジー計算が一般化されている [7].

2.2 O-加群の自由分解の間の鎖準同型の構成アルゴリズム

次のアルゴリズムは, 多項式環の場合にはよく知られており, Macaulay2 ではコマンド `incudedMap` に実装されている. このアルゴリズムは多項式環でのアルゴリズムを O-加群の場合にそのまま一般化したものである.

定理 **2.4.** M, N を有限生成左 O-加群とする. $M^\bullet, N^\bullet$ を M, N の自由分解, $\varphi : M \to N$ を O-加群の準同型とする. このとき, φ を誘導する鎖準同型 $\varphi^\bullet : M^\bullet \to N^\bullet$ を求めるアルゴリズムが存在する. すなわち, 下の図式を可換にする $\varphi^\bullet$ を構成することができる.

$$\begin{array}{lccccccccc}
M^\bullet: & \cdots & \longrightarrow & O^{m_2} & \xrightarrow{f_2} & O^{m_1} & \xrightarrow{f_1} & O^{m_0} & \xrightarrow{f_0} & M & \longrightarrow & 0 \\
 & & & \downarrow{\scriptstyle \varphi^2} & & \downarrow{\scriptstyle \varphi^1} & & \downarrow{\scriptstyle \varphi^0} & & \downarrow{\scriptstyle \varphi} & & \\
N^\bullet: & \cdots & \longrightarrow & O^{n_2} & \xrightarrow{g_2} & O^{n_1} & \xrightarrow{g_1} & O^{n_0} & \xrightarrow{g_0} & N & \longrightarrow & 0
\end{array}$$

Proof. f_1, g_1 を定める $m_1 \times m_0$ 行列, $k_1 \times k_0$ 行列をそれぞれ $P = [p_{ij}], Q = [q_{ij}]$ とする.

まず, φ^0 を求める. φ を定める行列を $\Phi = [\overline{\varphi_{ij}}]$ とすると, φ^0 は $\overline{\varphi_{ij}}$ の代表元を成分とする $m_0 \times k_0$ 行列 $\Phi^0 = [\varphi_{ij}]$ で与えられる. 次に, φ^1 を求める. 図式の可換性より, φ^1 は非斉次連立 1 次方程式 $P\Phi^0 = \Phi^1 Q$ を満たす $m_1 \times k_1$ 行列 Φ^1 で与えられる. [14, IV Theorem 4.1] により, この方程式には Φ^0 の定め方によらず解が存在する. この非斉次連立 1 次方程式の解は, Gröbner 基底を用いることで求めることができる. 多項式環の場合の方法は [11, p 214, Proposition 2.7] で述べられている. この方法は O-加群の場合にそのまま一般化できる. この操作を繰り返すことにより, $\varphi^\bullet$ が構成できる. □

鎖準同型の構成例については例 3.7 を参照.

3 平面曲線に対する twisted logarithmic cohomology 群の計算アルゴリズム

この章では $n = 2$ の場合を考える. 2 変数の場合に限定する理由を, この章の概略とともに述べておく. 2 変数の場合には, $\mathrm{Der}(-\log f)$ の自由基底の存在が保証されている. この事実から twisted logarithmic cohomology 群 があ る D-加群の積分と同型となることが示される (系 3.3). さらにこの同型を定理 2.4 を用いて構成することで, twisted logarithmic cohomology 群 の基底を計算する (ただし, 0 次のコホモロジー群は別の方法で求める). これが twisted logarithmic cohomology 群 を計算するアルゴリズムの概略である.

以下では, $D = D_2, x = x_1, y = x_2, \partial_x = \partial_1, \partial_y = \partial_2$ とする. $g \in R$ に対して $\frac{\partial g}{\partial x}, \frac{\partial g}{\partial y}$ をそれぞれ g_x, g_y で表すと約束する. $\otimes$ は D 上でのテンソル積を表すとする.

3.1 $(\Omega^\bullet(\log f), \nabla)$ の自由基底による表示

$n = 2$ のとき, 有限生成 reflexive R-加群は射影的加群となるから, Quillen-Suslin の定理 [11, p 197, Theorem 1.8] によりそれは自由加群となる. ゆえに $\mathrm{Der}(-\log f)$ (および $\Omega^1(\log f)$) はランク 2 の自由 R-加群となる. $\mathrm{Der}(-\log f)$ の基底 $\{\delta_1, \delta_2\}$ を

$$\delta_i = a_{i1}\partial_x + a_{i2}\partial_y \quad i = 1, 2$$

とすると, 齋藤の判定法 [25, 1.8] より, $\det((a_{ij})) = f$ が成り立つ. よって, $\{\delta_1, \delta_2\}$ の双対基底を ω_1, ω_2 とすると

$$\omega_1 = \frac{1}{f}(a_{22}dx - a_{21}dy),\ \omega_2 = \frac{1}{f}(-a_{12}dx + a_{11}dy)$$

となる. このとき, $\Omega^2(\log f)$ は $\omega_1 \wedge \omega_2 = \frac{dx \wedge dy}{f}$ を基底とするランク 1 の自由 R-加群となる.

[25, 1.5] より, $\mathrm{Der}(\log f)$ は交換子積 $[\cdot, \cdot]$ に関して閉じている. $[\delta_1, \delta_2]$ を δ_1, δ_2 の R-線型結合で表したときの係数を b_1, b_2 とする.

命題 3.1. 対数微分形式のなす複体 $(\Omega^\bullet(\log f), \nabla)$ は次の複体と同型となる.

$$\begin{array}{ccccc} \Omega^0(\log f) & \xrightarrow{\nabla} & \Omega^1(\log f) & \xrightarrow{\nabla} & \Omega^2(\log f) \\ {\scriptstyle \eta_0}\downarrow & & {\scriptstyle \eta_1}\downarrow & & {\scriptstyle \eta_2}\downarrow \\ R & \xrightarrow{\epsilon_1} & R^2 & \xrightarrow{\epsilon_2} & R \end{array}$$

ここで, 下の複体の境界写像は

$$\epsilon_1(g) = (\delta_1(g) + \sum_{i=1}^{s} \alpha_i \frac{\delta_1(f_i)}{f_i} g, \delta_2(g) + \sum_{i=1}^{s} \alpha_i \frac{\delta_2(f_i)}{f_i} g)$$

$$\epsilon_2(h_1, h_2) = \delta_1(h_2) - \delta_2(h_1) - b_1 h_1 - b_2 h_2 - \sum_{i=1}^{s} \alpha_i \frac{\delta_2(f_i)}{f_i} h_1 - \sum_{i=1}^{s} \alpha_i \frac{\delta_1(f_i)}{f_i} h_2$$

である. また, 複体の間の同型を与える鎖準同型は $\eta_0 = id$, $\eta_1(h_1\omega_1 + h_2\omega_2) = (h_1, h_2)$, $\eta_2(g\omega_1 \wedge \omega_2) = g$ $(g, h_1, h_2 \in R)$ で与えられる.

この命題はより一般のある条件を満たす複体 $\Omega^\bullet(\log f, d)$ で示されている定理 [4, Theorem 3.2.1] および系 [4, Corollary 3.2.2] の 2 変数, twisted 版である.

Proof. $\mathrm{Der}(-\log f) \subset \mathrm{Der}(-\log f_i)$ $(i = 1, \cdots, s)$ が成り立つことに注意を払えば, 後は [8, Remark 2.1] で d を ∇ に置き換えた議論によって主張が従う. □

上の命題に現れた複体

$$0 \to R \xrightarrow{\epsilon_1} R^2 \xrightarrow{\epsilon_2} R \to 0 \tag{1}$$

は右 D-加群の複体

$$0 \to D \xrightarrow{\bar{\epsilon_1}} D^2 \xrightarrow{\bar{\epsilon_2}} D \to 0 \tag{2}$$

に関手 $- \otimes D/(D\partial_x + D\partial_y)$ を施したものである. ただし, $P, Q_1, Q_2 \in D$ に対して境界写像は

$$\bar{\epsilon_1}(P) = (\delta_1 P + \sum_{i=1}^{s} \alpha_i \frac{\delta_1(f_i)}{f_i} P, \delta_2 P + \sum_{i=1}^{s} \alpha_i \frac{\delta_2(f_i)}{f_i} P)$$

$$\bar{\epsilon_2}(Q_1, Q_2) = \delta_1 Q_2 - \delta_2 Q_1 - b_1 Q_1 - b_2 Q_2 - \sum_{i=1}^{s} \alpha_i \frac{\delta_2(f_i)}{f_i} Q_1 - \sum_{i=1}^{s} \alpha_i \frac{\delta_1(f_i)}{f_i} Q_2$$

と定める. $I^{\log f} := \{-\delta_2 - b_1 - \sum_{i=1}^{s} \alpha_i \frac{\delta_2(f_i)}{f_i}, \delta_1 - b_2 - \sum_{i=1}^{s} \alpha_i \frac{\delta_1(f_i)}{f_i}\} \cdot D$ とおくと, 次の命題が成り立つ.

命題 3.2. $I^{\log f}$ はホロノミック右イデアルである.

Proof. $I^{\log f}$ の特性イデアル $\mathrm{in}_{(0,1)}(I^{\log f})$ には $\mathrm{in}_{(0,1)}(\delta_1)$, $\mathrm{in}_{(0,1)}(\delta_2)$ なる元が含まれる. これらが生成する $\mathbb{C}[x,y,\xi_x,\xi_y]$ のイデアルを J とすると, $\dim(\mathrm{in}_{(0,1)}(I^{\log f})) \leq \dim(J)$ が成り立つ. よって, $\dim(J) \leq 2$ であることを示せばよい. まず, δ_1, δ_2 が $\mathrm{Der}(-\log f)$ の自由基底であることから, $\mathrm{in}_{(0,1)}(\delta_1)$, $\mathrm{in}_{(0,1)}(\delta_2)$ は $\mathbb{C}[x,y,\xi_x,\xi_y]$-正則列となる. 実際, 正則列でないと仮定すると [4, Corollary 4.2.2] と同様の議論により, 齋藤の判定法に矛盾することになる. このとき, 可換環論でよく知られているように, $\mathrm{height}(J) = 2$ が成り立つ (例えば [15, p 159] を参照). さらに, 可換環論における一般論より $\mathrm{height}(J) + \dim(J) \leq \dim\mathbb{C}[x,y,\xi_x,\xi_y] = 4$ が成り立つから, $\dim(J) \leq 2$ となる. 従って, 主張が示された. □

$M^{\log f} := D/I^{\log f}$ とおいて (2) の右端に $M^{\log f}$ を加えた列を考える. この列の $(0,0,1,1)$-フィルターから定まる次数加群の列を考えると, それは正則列から定まる Koszul 複体となっている. したがって Koszul 複体のコホモロジー消滅定理 (例えば [15, 定理 16.5] を参照) と [3, p 46, Lemma 3.13] から (2) はホロノミック右 D-加群 $M^{\log f}$ の自由分解であることが分かる. 以上のことから次の系を得る.

系 3.3. Twisted logarithmic cohomology 群 はホロノミック右 D-加群 $M^{\log f}$ の積分と同型である. すなわち

$$H^\bullet(\Omega^\bullet(\log f), \nabla) \simeq H^\bullet((M^{\log f})^\bullet \otimes D/(D\partial_x + D\partial_y))$$

この系が twisted logarithmic cohomology 群の基底を計算するアルゴリズムの基礎を与える.

3.2 H^1, H^2 の計算

まず $H^2(\Omega^\bullet(\log f), \nabla)$ を計算するアルゴリズムを与える. 以下の議論は, [8, Algorithm 5.1] において d から定まる微分作用素 L_i たちを, ∇ から定まるものに置き換えたものである. 対数微分形式の定義から $\Omega^2(\log f) = \mathbb{C}[x,y]\frac{dx\wedge dy}{f}$ である. さらに $\Omega^1(\log f)$ は $\mathbb{C}[x,y]$-加群としてシヂジー加群 $\mathrm{Syz}(f_y, -f_x, f)$ と同型である. 実際, シヂジー (p,q,r) は対数 1-形式 $\omega = \frac{pdx+qdy}{f}$ を定める. Gröbner 基底を用いることで, $\mathrm{Syz}(f_y, -f_x, f)$ の生成元を求めることができる. その生成元を $(p_1,q_1,r_1), \cdots, (p_m,q_m,r_m)$ とする. 一方, 任意の $g \in \mathbb{C}[x,y]$ と任意の $\omega = \frac{pdx+qdy}{f} \in \Omega^1(\log f)$ に対して $\nabla(g\omega) = (L\bullet g)\frac{dx\wedge dy}{f}$ となる. ここで, L は

$$L = q\partial_x - p\partial_y + q_x - p_y - r + \sum_{j=1}^{m} \frac{\alpha_j}{f_j}(q(f_j)_x - p(f_j)_y)$$

なる微分作用素を表す. f_i は f の既約因子であるから, 簡単な計算より (p,q,r) はシヂジー加群 $\mathrm{Syz}((f_i)_y, -(f_i)_x, f_i)$ の元であることがわかる. これより L は多項式係数の微分作用素となる. 従って, $(p_1,q_1,r_1),\cdots,(p_m,q_m,r_m)$ に対応する微分作用素を $L_1,\cdots,L_m$ とすると $\nabla\Omega^1(\log f) = \sum_{i=1}^m L_i \bullet \mathbb{C}[x,y]\frac{dx\wedge dy}{f}$ となる. したがって,

$$\begin{aligned} H^2(\Omega^\bullet(\log f),\nabla) &= \mathbb{C}[x,y]/\sum_{i=1}^m L_i \bullet \mathbb{C}[x,y] \\ &\simeq D/(\{L_1\cdots L_m\}\cdot D + \sum_{i=1}^2 D\partial_i) \\ &\simeq (D/\{L_1\cdots L_m\}\cdot D)\otimes(D/\sum_{i=1}^2 D\partial_i) \end{aligned}$$

が成り立つ. 命題 3.2 より $\{L_1\cdots L_m\}\cdot D$ はホロノミックである. $H^2(\Omega^\bullet(\log f),\nabla)$ は右 $D/\{L_1\cdots L_m\}\cdot D$ の 0 次の積分と同型であるから, 右辺の基底は積分アルゴリズム [18] によって求まる. それを $\{c_i\}$ とすると, $\{c_i\frac{dx\wedge dy}{f}\}$ が $H^2(\Omega^\bullet(\log f),\nabla)$ の基底となる. また $M^{\log f} = D/\{L_1\cdots L_m\}\cdot D$ であることに注意する.

次に $H^1(\Omega^\bullet(\log f),\nabla)$ の基底を求めるアルゴリズムを与える. $M^\bullet$ を $M^{\log f}$ の自由分解, $N^\bullet$ を $M^{\log f}$ の $(1,1,-1,-1)$-adapted free resolution ([21, 22] を参照) とする. 定理 2.2 より, $M^\bullet$ と $N^\bullet$ の間には $id : M^{\log f} \to M^{\log f}$ を誘導する鎖準同型 $\varphi^\bullet : M^\bullet \to N^\bullet, \psi^\bullet : N^\bullet \to M^\bullet$ を Gröbner 基底で構成できる.

$$\begin{array}{lccccccccc} M^\bullet: & \cdots & \longrightarrow & D^{m_2} & \longrightarrow & D^{m_1} & \xrightarrow{[L_1,\cdots,L_m]\cdot} & D & \longrightarrow M^{\log f} \longrightarrow & 0 \\ & & & \psi_2\uparrow\downarrow\varphi^2 & & \psi^1\uparrow\downarrow\varphi^1 & & id\uparrow\downarrow id & & id\uparrow\downarrow id \\ N^\bullet: & \cdots & \longrightarrow & D^{k_2} & \longrightarrow & D^{k_1} & \xrightarrow{[\ell_1,\cdots,\ell_k]\cdot} & D & \longrightarrow M^{\log f} \longrightarrow & 0 \end{array}$$

ここで, $m_1 = m, k_1 = k$ であり, $\{\ell_1,\cdots,\ell_k\}$ は $\{L_1,\cdots,L_m\}$ の $(1,1,-1,-1)$-Gröbner 基底である. また, φ^1, ψ^1 はそれぞれ以下で定まる変換行列 P, Q である.

$$\begin{aligned} [L_1,\cdots,L_m] &= [\ell_1,\cdots,\ell_k]P \\ [\ell_1,\cdots,\ell_k] &= [L_1,\cdots,L_m]Q \end{aligned}$$

変換行列 P は, 割り算アルゴリズムを用いて $L_1,\cdots,L_m$ を Gröbner 基底 $\ell_1,\cdots,\ell_k$ の標準表示として表したときの商から求まる. また変換行列 Q は, $L_1,\cdots,L_m$ から $\ell_1,\cdots,\ell_k$ を求めるときの S 多項式を記憶しておくことで求まる. このとき, ホモロジー代数の一般的な議論から $\psi^\bullet\circ\varphi^\bullet$ は $id_{M^\bullet}$ に, $\varphi^\bullet\circ\psi^\bullet$ は $id_{N^\bullet}$ にそれぞれホモトピー同値であることがわかる (例えば, [14,

IV, Theorem 4.1] を参照). すなわち,

$$\psi^\bullet \circ \varphi^\bullet \simeq id_{M^\bullet},\ \varphi^\bullet \circ \psi^\bullet \simeq id_{N^\bullet}.$$

従って, 写像のテンソル積 $\varphi^\bullet \otimes id : M^\bullet \otimes (D/\sum_{i=1}^2 D\partial_i) \to N^\bullet \otimes (D/\sum_{i=1}^2 D\partial_i)$, $\psi^\bullet \otimes id : N^\bullet \otimes (D/\sum_{i=1}^2 D\partial_i) \to M^\bullet \otimes (D/\sum_{i=1}^2 D\partial_i)$ に対して

$$(\psi^\bullet \otimes id) \circ (\varphi^\bullet \otimes id) \simeq id_{M^\bullet} \otimes id,\ (\varphi^\bullet \otimes id) \circ (\psi^\bullet \otimes id) \simeq id_{N^\bullet} \otimes id$$

が成り立つ. ゆえに

$$(\psi^\bullet \otimes id)^* : H^\bullet(N^\bullet \otimes (D/\sum_{i=1}^2 D\partial_i)) \to H^\bullet(M^\bullet \otimes (D/\sum_{i=1}^2 D\partial_i))$$

はコホモロジー群の同型を与える. 特に 1 次のコホモロジー群の間の同型として, 行列 Q は $\mathbb{C}$-ベクトル空間の間の同型 $\overline{Q} : H^1(N^\bullet \otimes (D/\sum_{i=1}^2 D\partial_i)) \to H^1(M^\bullet \otimes (D/\sum_{i=1}^2 D\partial_i))$ を誘導する. ところで, $H^\bullet(N^\bullet \otimes (D/\sum_{i=1}^2 D\partial_i))$ の基底は積分アルゴリズム [22] によって求めることができる. ゆえに対応

$$\begin{array}{ccccc} H^1(N^\bullet \otimes (D/\sum_{i=1}^2 D\partial_i)) & \xrightarrow{\overline{Q}} & H^1(M^\bullet \otimes (D/\sum_{i=1}^2 D\partial_i)) & \longrightarrow & H^1(\Omega^\bullet(\log f)) \\ \cup & & \cup & & \cup \\ \begin{bmatrix} a_1^{(1)} \\ \vdots \\ a_k^{(1)} \end{bmatrix} & \longmapsto & Q \bullet \begin{bmatrix} a_1 \\ \vdots \\ a_k \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix} & \longmapsto & \sum_{i=1}^m b_i \dfrac{p_i dx + q_i dy}{f} \end{array} \tag{3}$$

によって, $H^1(N^\bullet \otimes (D/\sum_{i=1}^2 D\partial_i))$ の基底を $H^1(\Omega^\bullet(\log f), \nabla)$ の基底に移すことができる.

3.3 H^0 の計算

最後に $H^0(\Omega^\bullet(\log f), \nabla)$ の基底を求めるアルゴリズムを述べる. $\Omega^1(\log f)$ の生成元として自由基底を用いていないため, $H^0(M^\bullet \otimes (D/\sum_{i=1}^2 D\partial_i))$ と $H^0(\Omega^\bullet(\log f), \nabla)$ の間の同型を構成することは難しい. そこで, ここでは $H^0(\Omega^\bullet(\log f), \nabla)$ の基底の計算を微分方程式の多項式解を求めることに帰着させる. 対数微分形式の定義から $\Omega^0(\log f) = \mathbb{C}[x, y]$ である. $H^0(\Omega^\bullet(\log f), \nabla) = \mathrm{Ker}(\nabla : \Omega^0(\log f) \to \Omega^1(\log f))$ であるから, $g \in H^0(\Omega^\bullet(\log f), \nabla)$ をとると

$$\nabla(g) = ((\partial_x) \bullet g + \sum_{i=1}^s \alpha_i \frac{(f_i)_x}{f_i} g)dx + ((\partial_y) \bullet g + \sum_{i=1}^s \alpha_i \frac{(f_i)_y}{f_i} g)dy = 0$$

となる. ゆえに, $H^0(\Omega^\bullet(\log f), \nabla)$ は連立微分方程式

$$f(\partial_x) \bullet g + \sum_{i=1}^s \alpha_i f \frac{(f_i)_x}{f_i} g = 0, f(\partial_y) \bullet g + \sum_{i=1}^s \alpha_i f \frac{(f_i)_y}{f_i} g = 0 \tag{4}$$

の解空間である.

命題 3.4. 微分方程式系 (4) はホロノミックであり, ホロノミックランクは高々 1 である.

Proof. $p_1 := f\partial_x + \sum_i \alpha_i f \frac{(f_i)_x}{f_i}, p_2 := f\partial_y + \sum_i \alpha_i f \frac{(f_i)_y}{f_i}$ によって生成される D の左イデアルを I とおく.

$$\partial_y p_1 - \partial_x p_2 = \{f_y - \sum_i \alpha_i f \frac{(f_i)_y}{f}\}\partial_x - \{f_x - \sum_i \alpha_i f \frac{(f_i)_x}{f}\}\partial_y + (\sum_i \alpha_i f \frac{(f_i)_x}{f})_y - (\sum_i \alpha_i f \frac{(f_i)_y}{f})_x$$

であるから, $\mathrm{in}_{(0,1)}(I)$ には $f\xi_x, f\xi_y, \{f_y - \sum_i \alpha_i f \frac{(f_i)_y}{f}\}\xi_x - \{f_x - \sum_i \alpha_i f \frac{(f_i)_x}{f}\}\xi_y$ なる元が属する. ゆえに I の特性多様体は $V(f\xi_x, f\xi_y, \{f_y - \sum_i \alpha_i f \frac{(f_i)_y}{f}\}\xi_x - \{f_x - \sum_i \alpha_i f \frac{(f_i)_x}{f}\}\xi_y)$ なる代数多様体に含まれる. [8, Theorem 5.2] と同様の議論で, この代数多様体の次元は 2 以下であることが分かる. 従って, I はホロノミックである. (4) の形から I のホロノミックランクが高々 1 であることは明らかである. □

この命題から $\dim H^0(\Omega^\bullet(\log f), \nabla) \leq 1$ が従う. $H^0(\Omega^\bullet(\log f), \nabla)$ の次元は $H^0((M^{\log f})^\bullet \otimes D/(\sum_{i=1}^2 D\partial_i))$ の次元と等しいから, 先の積分アルゴリズムによって次元は求められる. さらに, この微分方程式の多項式解を求める方法はいろいろあるが, 例えば [23, Algorithm 2.4] によって求めることができる.

3.4 アルゴリズムとその実装

前節までの内容をまとめることにより, 2 次元の場合の twisted logarithmic cohomology 群を計算するアルゴリズムを得る.

アルゴリズム 3.5.
入力 : 0 でない既約多項式 $f_1, \cdots, f_s \in \mathbb{C}[x, y]$, パラメータ $\alpha_1, \cdots, \alpha_s \in \mathbb{Q}$
出力 : $H^i(\Omega^\bullet(\log f), \nabla)$ の次元と基底

1. $\mathbb{C}[x, y]$ 上の Gröbner 基底を用いて $\mathrm{Syz}(f_y, -f_x, f)$ の生成元を計算する. それを $(p_1, q_1, r_1), \cdots, (p_m, q_m, r_m)$ とする.

   $$L_i = q_i\partial_x - p_i\partial_y + (q_i)_x - (p_i)_y - r_i + \sum_{j=1}^{s} \frac{\alpha_j}{f_j}(q_i(f_j)_x - p_i(f_j)_y) \quad (5)$$

   によって微分作用素 $L_1, \cdots, L_m$ を定める.

2. 右イデアル $\{L_1, \cdots, L_m\} \cdot D$ の $(1, 1, -1, -1)$-Gröbner 基底を計算する. それを $\ell_i, \cdots, \ell_k$ とする. Gröbner 基底の計算過程において S-ペアを記憶しておくことで, $[\ell_1, \cdots, \ell_k] = [L_1, \cdots, L_m]Q$ なる変換行列 Q も計算する.

3. 右 D-加群 $M = D/\{\ell_1, \cdots, \ell_k\} \cdot D$ に積分アルゴリズムを適用して $H^i(M^\bullet \otimes D/\sum_{i=1}^2 D\partial_i)$ $(i = 0, 1, 2)$ の基底を計算する.

4. $H^2(M^\bullet \otimes D/\sum_{i=1}^2 D\partial_i)$ の基底を $\{c_i\}$ としたとき, $\{c_i \frac{dx \wedge dy}{f}\}$ を $H^2(\Omega^\bullet(\log f), \nabla)$ の基底として出力する.

5. $H^1(M^\bullet \otimes D/\sum_{i=1}^2 D\partial_i)$ の基底を計算し, 対応 (3) で移したものを $H^1(\Omega^\bullet(\log f), \nabla)$ の基底として出力する.

6. $\dim H^0(M^\bullet \otimes D/\sum_{i=1}^2 D\partial_i) = 1$ ならば, [23, Algorithm 2.4] によって (4) の多項式解を求める. その解を $H^0(\Omega^\bullet(\log f), \nabla)$ の基底として出力する. $H^0(M^\bullet \otimes D/\sum_{i=1}^2 D\partial_i) = 0$ ならば $H^0(\Omega^\bullet(\log f), \nabla) = 0$ である.

定理 3.6. アルゴリズム 5.1 のステップ 2 において右 D-加群 M に対する積分アルゴリズムは停止する.

Proof. 命題 3.2 より $I^{\log f}$ がホロノミックであることから主張が従う. □

アルゴリズム 3.5 の実装は数式処理システム Macaulay2 で行った. プログラムはパッケージ `twistedLogcohomology.m2` [33] に収められている. なお Macaulay2 では右 D-加群が扱えないため, アルゴリズムのステップ 1 で求めた微分作用素 $L_1, \cdots, L_m$ の共役作用素をとり, 左 D-加群として計算をさせている. 積分加群の基底は, Macaulay2 の D-module パッケージ [31] 内にある `DintegrationAll` コマンドで求めている. また微分方程式 (4) の多項式解は `polysols` コマンドで求めている.

例 3.7. $s = 2, f_1 = x, f_2 = x + y, \alpha_1 = -1, \alpha_2 = 0$ の場合. $f = x(x + y)$ に対して $\Omega^1(\log f)$ の生成元は $\omega_1 = \frac{(x+y)dx}{f}, \omega_2 = \frac{-ydx+xdy}{f}$ となる. この生成元から定まる微分作用素はそれぞれ

$$L_1 = -(x + y)\partial_y, \ L_2 = x\partial_x + y\partial_y - 1$$

となる. このとき $M^{\log f}$ の自由分解 $M^\bullet$ と $M^{\log f}$ の adapted $(1, 1, -1, -1)$-resolution $N^\bullet$, および $M^\bullet$ と $N^\bullet$ の間の同型を定める鎖準同型 $\varphi^\bullet$ は以下のようになる:

$$\begin{array}{lccccccccc}
M^\bullet: & 0 & \longrightarrow & D & \xrightarrow{f^2} & D^2 & \xrightarrow{f^1} & D & \longrightarrow M^{\log f} \longrightarrow & 0 \\
 & & & \psi^2 \uparrow \downarrow \varphi^2 & & \psi^1 \uparrow \downarrow \varphi^1 & & id \uparrow \downarrow id & & id \uparrow \downarrow id \\
N^\bullet: & 0 & \longrightarrow & D^2 & \xrightarrow{g^2} & D^3 & \xrightarrow{g^1} & D & \longrightarrow M^{\log f} \longrightarrow & 0
\end{array}$$

ただし,

$$f^1 = \begin{bmatrix} -(x+y)\partial_y & x\partial_x + y\partial_y - 1 \end{bmatrix},\ f^2 = \begin{bmatrix} -x\partial_x - y\partial_y + 1 \\ -(x+y)\partial_y \end{bmatrix},$$

$$g^1 = \begin{bmatrix} (x+y)\partial_y & x\partial_x + y\partial_y - 1 & -y\partial_x\partial_y + y\partial_y^2 - \partial_y \end{bmatrix},\ g^2 = \begin{bmatrix} \partial_x & -y\partial_x + y\partial_y - 2 \\ -\partial_y & 1 \\ 1 & -(x+y) \end{bmatrix},$$

$$\varphi^1 = \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 0 & 0 \end{bmatrix},\ \varphi^2 = \begin{bmatrix} x+y \\ 1 \end{bmatrix},\ \psi^1 = \begin{bmatrix} -1 & 0 & \partial_x \\ 0 & -1 & \partial_y \end{bmatrix},\ \psi^2 = \begin{bmatrix} 0 & 1 \end{bmatrix}$$

であり, 変換行列 Q は

$$Q = \begin{bmatrix} -1 & 0 & \partial_x \\ 0 & 1 & \partial_y \end{bmatrix}$$

となる. $M = D/\{\ell_1, \ell_2, \ell_3\} \cdot D$ に積分アルゴリズムを適用して $H^\bullet(M^\bullet \otimes D/\sum_{i=1}^2 D\partial_i)$ の基底を計算する. $H^2(M^\bullet \otimes D/\sum_{i=1}^2 D\partial_i)$ の基底は $\{y\}$ となり, $\{y\frac{dxdy}{f}\}$ が $H^2(\Omega^\bullet(\log f), \nabla)$ の基底となる. また $H^1(M^\bullet \otimes D/\sum_{i=1}^2 D\partial_i)$ の基底は $\left\{ \begin{bmatrix} 0 & -y & 0 \end{bmatrix}^T, \begin{bmatrix} -2x & 0 & 0 \end{bmatrix}^T \right\}$ となる. よって, $H^1(\Omega^\bullet(\log f), \nabla)$ の基底は $\left\{ \frac{y(ydx - xdy)}{f}, \frac{2x(x+y)dx}{f} \right\}$ となる. さらに $\dim H^0(M^\bullet \otimes D/\sum_{i=1}^2 D\partial_i) = 1$ となり, (4) の多項式解を求めると x となる. ゆえに, $H^0(\Omega^\bullet(\log f), \nabla)$ の基底は $\{x\}$ となる.

Macaulay2 で twisted logarithmic cohomology 群の基底を求めるコマンドは `twistedLogCohomology2` である. Twisted logarithmic cohomology の基底の計算に必要なのは変換行列のみであり, その計算は `twistedLogCohomology2` の内部で行っている. そのため, D の場合の鎖準同型を構成するアルゴリズムの実装は行っていない.

```
Macaulay 2, version 1.1.99

i1 : loadPackage "Dmodules"

o1 = Dmodules

o1 : Package

i2 : load "twistedLogCohomology2.m2"

i3 : R = QQ[x,y];

i4 : twistedLogCohomology2({x,x+y},{-1,0})
Warning: not a generic weight vector.  Could be difficult...

o4 = HashTable{BFunction => (s - 1)                                    }
                                    1
```

```
CohomologyGroups => HashTable{0 => QQ }
                                  2
                            1 => QQ
                                   1
                            2 => QQ
LogBasis => HashTable{0 => | x |                    }
                      1 => | y2dx-xydy 2x2dx+2xydx |
                      2 => | ydxdy |
                           1                        2                       1
OmegaRes => (QQ[x, y, dx, dy])  <-- (QQ[x, y, dx, dy])  <-- (QQ[x, y, dx, dy])  <-- 0

           0                        1                       2                       3
PreCycles => HashTable{0 => | -x2-xy |}
                            | -x     |
                       1 => | 0  -2x |
                            | -y 0   |
                            | 0  0   |
                       2 => | y |
                              1                      3                       2
VResolution => (QQ[x, y, dx, dy])  <-- (QQ[x, y, dx, dy])  <-- (QQ[x, y, dx, dy])  <-- 0

              0                       1                       2                       3

o4 : HashTable
```

Twisted logarithmic cohomology の基底はハッシュテーブル LogBasis で求められている. ただし, H^1, H^2 の基底は分子のみ出力される.

例 **3.8.** $s = 1, f = f_1 = (x^3+y^4+xy^3)(x^2+y^2), \alpha = \alpha_1 = 0$ の場合. この例は Castro-高山アルゴリズムにおいて, Quillen-Suslin アルゴリズムを用いなければ logarithmic cohomology 群が求められなかった例である ([8, Example 1.2.(b), Example 4.1.] を参照). (1) $H^0(\Omega^\bullet(\log f), \nabla) \simeq \mathbb{C} \cdot 1$ である. (2) $H^1(\Omega^\bullet(\log f), \nabla)$ は次の 3 個の元で張られる.

- $\omega_1 = \dfrac{1}{f}(10x^2y^3dx+21xy^4dx+y^5dx-15x^3y^2dy-26x^2y^3dy-6xy^4dy-5y^5dy+6x^3ydx-15x^2y^2dx-6x^4dy+15x^3ydy)$
- $\omega_2 = \dfrac{1}{f}(143x^2y^3dx+303xy^4dx-10y^5dx-255x^3y^2dy-415x^2y^3dy-102xy^4dy-112y^5dy+75x^3ydx-255x^2y^2dx-75x^4dy+228x^3ydy-27x^4dx)$
- $\omega_3 = \dfrac{1}{f}(50x^2y^3dx+78xy^4dx-22y^5dx-75x^3y^2dy-103x^2y^3dy-3xy^4dy-25y^5dy+3x^3ydx-75x^2y^2dx-3x^4dy+75x^3ydy)$

(3) $H^2(\Omega^\bullet(\log f), \nabla)$ は次の 7 個の元で生成される.

$$\frac{dx\wedge dy}{f}, -x\frac{dx\wedge dy}{f}, y^2\frac{dx\wedge dy}{f}, x^2y^2\frac{dx\wedge dy}{f}, xy^2\frac{dx\wedge dy}{f}, x^3y\frac{dx\wedge dy}{f}, y^4\frac{dx\wedge dy}{f}$$

パッケージ twistedLogCohomology2.m2 のコマンド twistedLogCohomology2 を 20G メモリの Intel Xeon(R) (2.33GHz) で実行したときの実行時間は, 0.46 秒であった.

```
i5 : time twistedLogCohomology2({x^3+y^4+x*y^3,x^2+y^2},{0,0})
Warning: not a generic weight vector.  Could be difficult...
     -- used 0.460028 seconds

                                                                 11      10
o5 = HashTable{BFunction => (s)(s - 4)(s - 2)(s - 1)(s - --)(s - --)
                                                                  3       3
                                       1
CohomologyGroups => HashTable{0 => QQ }
                                       3
                                  1 => QQ
                                       7
                                  2 => QQ
LogBasis => HashTable{0 => |1|
                      1 => | -177120x2y3dx-371952xy4dx-17712y5dx+265680x3y2dy+460
512x2y3dy+106272xy4dy+88560y5dy-106272x3ydx+265680x2y2dx+106272x4dy-265680x3ydy
/* \omega_1 を -17712 倍した元 */
926640x2y3dx+1963440xy4dx-64800y5dx-1652400x3y2dy-2689200x2y3dy-660960xy4dy-725760
y5dy-174960x4dx+486000x3ydx-1652400x2y2dx-486000x4dy+1477440x3ydy
/* \omega_2 を 6480 倍した元 */
324000x2y3dx+505440xy4dx-142560y5dx-486000x3y2dy-667440x2y3dy-19440xy4dy-162000y5d
y+19440x3ydx-486000x2y2dx-19440x4dy+486000x3ydy
/* \omega_3 を 6480 倍した元 */|
                      2 => | dxdy -xdxdy y2dxdy x2y2dxdy xy2dxdy x3ydxdy y4dxdy |

VResolution => (QQ[x, y, dx, dy])  <-- (QQ[x, y, dx, dy])  <-- (QQ[x, y, dx, dy])

               0                        1                        2
               <-- (QQ[x, y, dx, dy])

                   3

o5 : HashTable
```

例 **3.9.** 下の表は $f = x^p+y^q+xy^q$ に対する Twisted logarithmic cohomology 群をパラメータが $\alpha = 0, \frac{1}{2}$ の場合に, コマンド twistedLogCohomology2 でそれぞれ計算したときの次元と計算時間を表したものである. 次元は左から順に H^2, H^1, H^0 の次元を表す. 計算時間の測定は Intel Xeon E5410(2.33GHz) で行った.

The computation time of `twistedLogCohomology2` (seconds)

表 1: Parameter $\alpha = 0$

(p, q)	Dimensions	Timing
$(6, 7)$	$(4, 1, 1)$	2.96
$(6, 8)$	$(5, 1, 1)$	28.68
$(6, 9)$	$(6, 1, 1)$	1752.05
$(6, 10)$	—	> 3 hours

表 2: Parameter $\alpha = 1/2$

(p, q)	Dimensions	Timing
$(6, 7)$	$(4, 0, 0)$	0.20
$(6, 8)$	$(5, 0, 0)$	3.58
$(6, 9)$	$(6, 0, 0)$	483.71
$(6, 10)$	$(7, 0, 0)$	570.41

コマンド `twistedLogCohomology2` の計算で, $(p, q) = (6, 7), (6, 8)$ の場合と $(p, q) = (6, 9), (6, 10)$ の場合とで計算時間に大きな差が現れるのは, 積分アルゴリズムにおける b-関数の計算に大変な時間が費やされるためである.

我々はまた, 数式処理システム Risa/Asir 上で twisted logarithmic cohomology 群の middle cohomology 群の基底のみを計算するプログラムを実装した. Risa/Asir には, 野呂 [17] による効率のよい b-関数計算のアルゴリズムが実装されている. 我々のプログラムはパッケージ `ns_twistedlog.rr` [34] に収められているコマンド `ns_twistedlog.twisted_log_cohomology` で実行できる. 下の表はこのコマンドで同じ多項式 $f = x^p + y^q + xy^q$ に対する twisted logarithmic cohomology 群 $H^2(\Omega^\bullet(\log f), \nabla)$ を計算したときの次元と計算時間を表したものである. 計算時間の測定は Intel Xeon E5410(2.33GHz) で行った. Middle cohomology 群のみの計算ではあるが, コマンド `twistedLogCohomology2` よりも大きな例が速く計算できることが分かる.

The computation time of `ns_twistedlog.twisted_log_cohomology` (seconds)

表 3: Parameter $\alpha = 0$

(p, q)	$\mathrm{Dim} H^2$	Timing
$(30, 31)$	28	5.54
$(30, 32)$	29	6.03
$(30, 33)$	30	6.86
$(30, 40)$	37	16.63
$(30, 50)$	42	28.71
$(30, 60)$	57	122.54

表 4: Parameter $\alpha = 1/2$

(p, q)	$\mathrm{Dim} H^2$	Timing
$(30, 31)$	28	4.85
$(30, 32)$	29	5.30
$(30, 33)$	30	6.26
$(30, 40)$	37	15.31
$(30, 50)$	42	27.55
$(30, 60)$	57	177.92

4 制限加群 $D/I+\sum_{i=1}^m x_iD$ における割り算アルゴリズム

この章ではホロノミック左加群 D/I の $x_1=\cdots=x_m=0$ への制限における割り算アルゴリズムについて説明する. これは大阿久 [18] で述べられていることではあるが, 我々のアルゴリズムを正確に記述するために, この章にまとめておく.

4.1 左 D'-加群 $(D')^r\bigoplus D^s$ における割り算アルゴリズム

$D=K\langle x_1,\cdots,x_n,\partial_1,\cdots,\partial_n\rangle, D'=K\langle x_{m+1},\cdots,x_n,\partial_{m+1},\cdots,\partial_n\rangle$ とする. 左 D'-加群 $(D')^r\bigoplus D^s$ を考える. $(0)^r\bigoplus D^s$ に属する単項式は $(D')^r\bigoplus D^s$ の任意の単項式で割りきれないと約束する. $(D')^r$ の基底を $e_1,\cdots,e_r$, D^s の基底を $e_{-1},\cdots,e_{-s}$ とする. $(D')^r$ の項順序 $<$ をひとつ固定する. $p=\sum_{i=1}^r p_ie_i+\sum_{j=1}^s p_{-j}e_{-j}\in(D')^r\bigoplus D^s$ に対して, $\sum_{i=1}^r p_ie_i\neq 0$ のとき $\mathrm{in}_<(p)=\mathrm{in}_<(\sum_{i=1}^r p_ie_i)$ と定義する. $(0)^r\bigoplus D^s$ の元が余りに現れたときには 0 とみなすことで, 自由加群の場合と同様にして $(D')^r\bigoplus D^s$ における割り算アルゴリズムを示すことができる. これより $(D')^r\bigoplus D^s$ 上で Gröbner 基底の理論を展開することができる. 適用例は後述の例 4.2 を参照のこと.

4.2 制限加群 $D/I+\sum_{i=1}^m x_iD$ における割り算アルゴリズム

$w=(w_1,\cdots,w_m,\cdots,w_{m+1},\cdots,w_n)$ を重みベクトルで $w_i>0$ $(i=1,\cdots,m)$, $w_i=0$ $(i=m+1,\cdots,n)$ を満たすとする. I を D のホロノミックイデアルとし, $b(s)$ を I の $(-w,w)$ に関する b-関数, s_0 をその最大非負整数根とする. I の $(-w,w)$-Gröbner 基底を $\{g_1,\cdots,g_k\}$ とする. また $m_i=\mathrm{ord}_{(-w,w)}(g_i)$ $(i=1,\cdots,k)$ とする. このとき, [18, Theorem 5.7], [26, Theorem 5.2.6] により任意の $p\in D$ は以下の表示を持つ.

$$p=\sum_{i=1}^k c_ig_i+\sum_{j=1}^m x_jr_j+\sum_{\partial^\beta\in\mathcal{B}_{s_0},a_\beta\in D'}a_\beta\partial^\beta \tag{6}$$

ただし $\mathrm{ord}_{(-w,w)}(p)=s$ とすると, c_i は $c_i=\sum_{\partial^{\alpha_i}\in\mathcal{B}_{s-m_i}}c_{\alpha_i}\partial^{\alpha_i}$, $c_{\alpha_i}\in D'$ なる形をしている. $\sum_{\partial^\beta\in\mathcal{B}_{s_0},a_\beta\in D'}a_\beta\partial^\beta$ を p の $\bmod I+\sum_{j=1}^m x_jD$ に関する正規形といい, $p\equiv\sum_{\partial^\beta\in\mathcal{B}_{s_0},a_\beta\in D'}a_\beta\partial^\beta\ (\bmod I+\sum_{j=1}^m x_jD)$ と表す.

F_s の任意の元は $\sum_{\partial^\beta\in\mathcal{B}_s,a_\beta\in D'}a_\beta\partial^\beta+\sum_{j=1}^m x_jr_j$ なる形に書けるので $F_s\subset(D')^{n_s}\bigoplus D$ と見なせる.

定理 4.1. ([18]) I をホロノミックイデアルとすると, $p\in D$ の正規形を計算するアルゴリズムが存在する.

Proof. $\{\partial^{\alpha_i} g_i \mid i = i, \cdots, k, \partial^{\alpha_i} \in \mathcal{B}_{s-m_i}\}$ で生成される $(D')^{n_s} \bigoplus D$ の部分加群と p に対して, $(D')^{n_s} \bigoplus D$ における割り算アルゴリズムを適用する. [18, Lemma 5.1] により, この余りが p の正規形になることが分かる. □

Fourier 変換 [26, p 229] で積分加群を制限加群に変換することにより, 積分加群における割り算アルゴリズムも得られる.

例 **4.2.** $n = 2, m = 1, (-w, w) = (-1, 0, 1, 0)$ とする.

$L_1 = 2x_1\partial_1\partial_2 + 2x_2\partial_2^2 + 2x_2\partial_2 - \partial_2$, $L_2 = 2x_2\partial_1\partial_2 + 2x_2\partial_2^2 + 2x_2\partial_2 - \partial_2$,
$L_3 = -2x_1\partial_1^2 + 2x_2\partial_2^2 - 2x_1\partial_1 + 2x_2\partial_2 + \partial_1 - \partial_2$

とすると L_1, L_2, L_3 が生成するイデアル I はホロノミックである.

$$p = -2x_1x_2\partial_1^2 + x_1\partial_1\partial_2^2 + 2\partial_2^3 + x_2\partial_1$$

とする. $x_1 = 0$ への制限加群における p の正規形を求める.

$$L_4 = -2x_1x_2\partial_2^2 + 2x_2^2\partial_2^2 - 2x_1x_2\partial_2 + 2x_2^2\partial_2 + x_1\partial_2 - x_2\partial_2$$

とおくと $\{L_1, L_2, L_3, L_4\}$ が I の $(-w, w)$-Grönber 基底となることが分かる. $\mathrm{ord}_{(-w,w)}(p) = 1$ であり, $m_i = \mathrm{ord}_{(-w,w)}(L_i)$ とおくと $(m_1, m_2, m_3, m_4) = (0, 1, 1, 0)$ である. $D_1^{n_s} = D_1^2$ の基底を $e_1 = 1, e_2 = \partial_1$ と表す. $<$ を $e_1 < e_2$ と D_1 の全次数辞書式順序 $(x_1 < \partial_1)$ から定まる POT 順序とする. $\{\partial_1^{\alpha_1}\partial_2^{\alpha_2} L_i \mid i = 1, 2, 3, 4, \alpha_1 + \alpha_2 \leq 1 - m_i\}$ が生成する $D_1^2 \bigoplus D_2 = D_1e_1 \bigoplus D_1e_2 \bigoplus D_2e_{-1}$ の部分加群の $<_w$-Grönber 基底は

$$g_1 = L_1 = (2x_2\partial_2^2 + 2x_2\partial_2 - \partial_2)e_1 + 2x_1\partial_1\partial_2 e_{-1},$$
$$g_2 = L_3 - L_1 = e_2 - 2x_1(\partial_1^2 + \partial_1\partial_2 + \partial_1)e_{-1}$$

となる. $p = 2x_2e_2 + 2x_2\partial_2^3 e_1 - 2x_1(x_2\partial_1^2 - \partial_1\partial_2)e_{-1}$ と見なせる. このとき,

$$\begin{aligned} p &= (\partial_2 - 1)g_1 + x_2g_2 + 2x_1(x_2\partial_1\partial_2 + x_2\partial_1 + \partial_1\partial_2) + 2x_2\partial_2 - \partial_2^2 - 3\partial_2 \\ &= (-x_2 + \partial_2 - 1)L_1 + x_2L_3 + 2x_1(x_2\partial_1\partial_2 + x_2\partial_1 + \partial_1\partial_2) + 2x_2\partial_2 - \partial_2^2 - 3\partial_2 \end{aligned}$$

となる. よって p の正規形は $2x_2\partial_2 - \partial_2^2 - 3\partial_2$ である.

5 多項式ベキの積分から定まるある超幾何積分の満たす差分方程式系への応用

5.1 $H^2(\Omega^\bullet(\log f), \nabla)$ の基底から定まる超幾何積分の場合

次の形の超幾何積分

$$p(\alpha_1, \cdots, \alpha_s) = \int_C \prod_{i=1}^{s} f_i(x, y)^{\alpha_i - 1} dxdy \tag{7}$$

を考える．$\Phi = \prod_{i=1}^{s} f_i(x,y)^{\alpha_i}, f = \prod_{i=1}^{s} f_i(x,y)$ とおくと (7) は $p(\alpha) = \int_C \Phi \frac{dxdy}{f}$ と書ける．積分路 C は $\mathbb{R}^2 \setminus \{f_i = 0 \mid i = 1, \cdots, s\}$ 上の 2-単体の $\mathbb{C}$-係数の形式的有限和であるとする．多価関数 Φ の分枝は C の各連結成分上で一つ固定しておく．また, パラメータに関する差分と積分との順序交換が可能であると仮定する．

差分作用素 E_{α_i} を $E_{\alpha_i} \bullet p(\alpha_1, \cdots, \alpha_i, \cdots, \alpha_s) = p(\alpha_1, \cdots, \alpha_i + 1, \cdots, \alpha_s)$ と定める．この節では超幾何積分 (7) の満たす非斉次差分方程式系を求めるアルゴリズムを与える．このアルゴリズムの実装は数式処理システム Risa/Asir で行った．ただし, 実装したプログラムはパラメータが $\alpha_1, \cdots, \alpha_s \notin \mathbb{Z}$ である場合にのみ対応している．プログラムはパッケージ `ns_twistedlog.rr` [34] に収められている．なお, プログラムの実装にはパッケージ `nk_restriction.rr` [32] を使用した．

外微分 $\nabla = d + d\log\Phi\wedge$ から定まる twisted logarithmic cohomology 群 $H^2(\Omega^\bullet(\log f), \nabla)$ の基底を $c_1 \frac{dxdy}{f}, \cdots, c_m \frac{dxdy}{f}$ とする．このとき $f_i^\ell \frac{dxdy}{f}$ は $\bmod \nabla$ で $c_1 \frac{dxdy}{f}, \cdots, c_m \frac{dxdy}{f}$ たちの $\mathbb{C}$ 線型結合で表される:

$$f_i^\ell \frac{dxdy}{f} \equiv a_{i1} c_1 \frac{dxdy}{f} + \cdots + a_{im} c_m \frac{dxdy}{f} \pmod{\nabla} \ (i = 1, \cdots, s, \ \ell = 0, \cdots, m).$$

これより, アルゴリズム 5.1 のステップ 2 で定めた ℓ_k に対して

$$f_i^\ell \frac{dxdy}{f} + \sum_k \ell_k \bullet g_k \frac{dxdy}{f} = a_{i1} c_1 \frac{dxdy}{f} + \cdots + a_{im} c_m \frac{dxdy}{f} \tag{8}$$

が成り立つ．(8) から $c_1 \frac{dxdy}{f}, \cdots, c_m \frac{dxdy}{f}$ を消去すると $f_i^\ell \frac{dxdy}{f}$ たちの間の非斉次一次関係式を得る:

$$\sum_{i,\ell} b_{i\ell}^{(j)} f_i^\ell \frac{dxdy}{f} + \sum_k \ell_k^{(j)} \bullet h_k^{(j)} \frac{dxdy}{f} = 0 \ (j = 1, \cdots, n).$$

両辺に Φ を掛けて C 上で積分すると

$$\sum_{i,\ell} b_{i\ell}^{(j)} \int_C \Phi f_i^\ell \frac{dxdy}{f} + \sum_k \int_C (\ell_k^{(j)} \bullet h_k^{(j)}) \Phi \frac{dxdy}{f} = 0.$$

となる．$E_{\alpha_i} \bullet p(\alpha) = \int_C \Phi f_i \frac{dxdy}{f}$ であるから

$$\sum_{i,\ell} b_{i\ell}^{j} E_{\alpha_i}^{\ell} \bullet p + \sum_k \int_C (\ell_k^{(j)} \bullet h_k^{(j)}) \Phi \frac{dxdy}{f} = 0 \ (j = 1, \cdots, n) \tag{9}$$

が成り立つ．これが $p(\alpha)$ が満たす非斉次差分方程式系である．

アルゴリズム 5.1.
入力：$f_1, \cdots, f_s \in \mathbb{C}[x,y]$, パラメータ $\alpha_1, \cdots, \alpha_s$
出力：$p(\alpha) = \int_C \prod_{i=1}^{s} f_i^{\alpha_i - 1} dxdy$ が満たす非斉次差分方程式

1. アルゴリズム 3.5 のステップ 4 までを行い, $H^2(\Omega^\bullet(\log f), \nabla)$ の基底 $c_1, \cdots, c_m$ を求める.

2. f_i^ℓ の全次数を $n_{i\ell}$ とする $(i = 1, \cdots, s, \ell = 0, \cdots, m)$. アルゴリズム 3.5 のステップ 2 で得た Gröbner 基底 $\{\ell_1, \cdots, \ell_k\}$ に対して, $\mathrm{ord}_{(1,1,-1,-1)}(\ell_i) = m_i$ とする.

3. $\{\ell_i \bullet x^\alpha y^\beta \mid i = 1, \cdots, k, \alpha + \beta \leq n_{i\ell} - m_i\}$ を用いて f_i^ℓ を $c_1, \cdots, c_m$ と $\ell_i \bullet x^\alpha y^\beta$ の線型結合で表す. (積分加群における割り算アルゴリズムを用いる.)

4. ステップ 3 で得た関係式から $c_1, \cdots, c_m$ を消去する. それを $\sum_{i,\ell} b_{i\ell}^{(j)} f_i^\ell + \sum_{i=1}^k \ell_i^{(j)} \bullet h_i^{(j)} = 0$ とする.

5. $\sum_{i,\ell} b_{i\ell}^{(j)} E_{\alpha_i}^\ell \bullet p + \sum_{i=1}^k \int_C (\ell_i^{(j)} \bullet h_i^{(j)}) \Phi \frac{dxdy}{f} = 0$ が $p(\alpha)$ の満たす差分方程式系.

定義 5.2. $E_s = \mathbb{C}(\alpha_1, \cdots, \alpha_s)\langle E_{\alpha_1}, \cdots, E_{\alpha_s}\rangle$ を s 変数差分作用素環とする. E_s の左イデアル I が次の条件を満たすとき 0 次元であるという:
任意の $1 \leq i \leq s$ に対して $\sum_\ell p_\ell(\alpha) E_{\alpha_i}^\ell \neq 0$ $(p_\ell(\alpha) \in \mathbb{C}(\alpha_1, \cdots, \alpha_s))$ なる元が I に存在する.

アルゴリズム 5.1 から得られる差分方程式系が斉次の場合には, 次の定理が成り立つ.

定理 5.3. アルゴリズム 5.1 から得られる差分作用素が生成する E_s の左イデアルは 0 次元である. すなわち, 各変数に関して消去法を行うことで

$$\sum_\ell c_{i\ell} E_{\alpha_i}^\ell \bullet p = 0 \ (i = 1, \cdots, s)$$

なる形の差分方程式を得ることができる.

Proof. 仮定より $\dim H^2(\Omega^\bullet(\log f)) = m$ であるから, $1, f_i, \cdots, f_i^m$ は mod ∇ に関して線型従属である. よってアルゴリズム 5.1 のステップ 4 において $1, f_i, \cdots, f_i^m$ の間の関係式を計算することができる. それを $\sum_\ell c_{i\ell} f_i^\ell = 0$ と表す. このとき $\sum_\ell c_{i\ell} E_{\alpha_i}^\ell \bullet p = 0$ $(i = 1, \cdots, s)$ が成り立つ. □

注意 5.4. 積分路 C が以下の条件を満たす場合, アルゴリズム 5.1 の差分方程式系は斉次となる.

1. 任意の対数 1-形式 ω に対して, $\int_{\partial C} \Phi\omega = 0$ が成り立つ.

2. パラメータ α_i を整数だけずらしても, 1 の条件が成り立つ.

f_i が全て一次式であり, かつ $f_i = 0$ によって定まる超平面たちが一般の位置にある場合, パラメータが条件 $\alpha_1, \cdots, \alpha_s \notin \mathbb{Z}$, $\sum_{i=1}^{s} \alpha_i \notin \mathbb{Z}$ を満たせば, このような C が具体的に構成できることが知られている ([2, p 124 $\sim$ p 133] を参照).

注意 5.5. アルゴリズム 5.1 のステップ 2 において一つの f_i だけを考えることにより, E_{α_i} に関する差分方程式だけを求めることができる. この方法はアルゴリズム 5.1 で得られた結果から消去法を行うよりも, 効率のよい計算である.

例 5.6. $p(a,b,c) = \int_C x^{a-1}y^{b-1}(1-x-y)^{c-1}dxdy$ とする. ただし $a, b, c \notin \mathbb{Z}$ とする. アルゴリズム 3.5 より, $H^2(\Omega^\bullet(\log f), \nabla) = \mathbb{C}\frac{dxdy}{f}$ となる. アルゴリズム 5.1 より, 対数 2-形式 $\frac{xdxdy}{f}$, $\frac{ydxdy}{f}$, $\frac{(1-x-y)dxdy}{f}$ は次の関係式を満たす:

$$(a+b+c)\frac{xdxdy}{f} - a\frac{dxdy}{f} + \ell \bullet h_1 \frac{dxdy}{f} = 0, (a+b+c)\frac{ydxdy}{f} - b\frac{dxdy}{f} + \ell \bullet h_2 \frac{dxdy}{f} = 0,$$
$$(a+b+c)\frac{(1-x-y)dxdy}{f} - c\frac{dxdy}{f} + \ell \bullet h_3 \frac{dxdy}{f} = 0.$$

ただし,

$$\ell = -y(y-1)\partial_x\partial_y + y(y+1)\partial_y^2 - \{bx + (b+c)y - b\}\partial_x + \{(a+b+c)y - b\}\partial_y,$$
$$h_1 = \frac{1}{b}\{(a+b+c)x + cy\}, h_2 = -y, h_3 = -\frac{1}{b}\{(a+b+c)x + (b+c)y\}$$

である. よって p は次の差分方程式を満たす:

$$\{(a+b+c)E_a - a\} \bullet p + \int_C (\ell \bullet h_1)\Phi\frac{dxdy}{f} = 0, \{(a+b+c)E_b - b\} \bullet p + \int_C (\ell \bullet h_2)\Phi\frac{dxdy}{f} = 0,$$
$$\{(a+b+c)E_c - c\} \bullet p + \int_C (\ell \bullet h_3)\Phi\frac{dxdy}{f} = 0.$$

Risa/Asir で超幾何積分 (7) の差分方程式系を計算するコマンドは `ns_twistedlog.difference_equation` である. 非斉次項の計算を行うにはオプション `inhomo` を用いる.

```
[1630] load("nk_restriction.rr");
[1859] load("ns_twistedlog.rr");
[1875] ns_twistedlog.difference_equation([x,y,1-x-y],[a,b,c],[x,y]|inhomo=1);
 -- nd_weyl_gr :0sec(0.000881sec)
-- weyl_minipoly_by_elim :0.004sec + gc : 0.004sec(0.006541sec)
Order : 1
[[(-ea+1)*b*a-ea*b^2-ea*c*b,[((-y^2+y)*dy-b*x+(-b-c)*y+b)*dx+(y^2-y)*dy^2+((a+
b+c)*y-b)*dy,(a+b+c)*x+(b+c)*y]],[-eb*a+(-eb+1)*b-eb*c,[((-y^2+y)*dy-b*x+(-b-c
)*y+b)*dx+(y^2-y)*dy^2+((a+b+c)*y-b)*dy,-y]],[-ec*b*a-ec*b^2+(-ec+1)*c*b,[((-y
^2+y)*dy-b*x+(-b-c)*y+b)*dx+(y^2-y)*dy^2+((a+b+c)*y-b)*dy,(-a-b-c)*x-c*y]]]
```

非斉次項を計算するオプションを指定した場合,

$$[[\text{E1}, [\text{OP1,P1}], [\text{OP2,P2}], \cdots], [\text{E2}, [\text{OP3,P3}], \cdots], \cdots]$$

のようなリストのリストが返ってくる. E1,E2,$\cdots$ は差分作用素, OP1,OP2,$\cdots$ は微分作用素, P1,P2,$\cdots$ は多項式である. OP1,$\cdots$, P1,$\cdots$ はそれぞれ (9) の $\ell_k^{(j)}, h_k^{(j)}$ に対応するものである.

注意 **5.7.** ns_twistedlog.difference_equation では, パラメータの入力として次のことを仮定している.

1. 多項式のベキの部分は有理数であるか, または $a+1/2$ のような (パラメータを表す 1 文字)+(有理数) の形をしている.
2. パラメータは全て相異なる.
3. パラメータを表す変数の係数は 1 である.

従って, 以下のような入力に対しては正しく動かない.

```
[289] ns_twistedlog.difference_equation([x,y,1-x-y],[a,b,a-b],[x,y]);
-- nd_weyl_gr :0.004114sec(0.005315sec)
-- weyl_minipoly_by_elim :0.002413sec(0.01103sec)
Order : 1
[ea,-eb,-1] /*間違った出力*/

[290] ns_twistedlog.difference_equation([x,y,1-x-y],[-a,-b,-c],[x,y]);
-- nd_weyl_gr :0.003542sec(0.004116sec)
-- weyl_minipoly_by_elim :0.007714sec(0.008479sec)
Order : 1
[(-ea+1)*a-ea*b-ea*c,eb*a+(eb-1)*b+eb*c,-ec*a-ec*b+(-ec+1)*c] /*間違った出力*/
```

次の注意に出てくるコマンド ns_twistedlog.twisted_log_cohomology や, 後に出てくる微分方程式系を計算するコマンド twistedlog.differential_equation に対しては, このような注意は払わなくてよい.

注意 **5.8.** ns_twistedlog.difference_equation ではパラメータを不定元とみなして計算している. 例えば, ns_twistedlog.twisted_log_cohomology は twisted logarithmic cohomology 群の middle cohomology 群の基底のみを計算するコマンドであるが, 多項式のベキが整数の場合とパラメータの場合では, 次のように出力結果が異なる場合がある.

```
[1971] ns_twistedlog.twisted_log_cohomology([x,y,1-x-y],[-1,-2,-3],[x,y]);
-- nd_weyl_gr :0.00172sec(0.001897sec)
-- weyl_minipoly_by_elim :0.004114sec(0.004109sec)
-- generic_bfct_and_gr :0.007336sec(0.007513sec)
generic bfct : [[-1,1],[s,1],[s-7,1]]
S0 : 7
B_{S0} length : 36
-- fctr(BF) + base :0.02426sec(0.02503sec)
dimension : 3
[y^2*x^5,y^7,1]
```

```
[1972] ns_twistedlog.twisted_log_cohomology([x,y,1-x-y],[a,b,c],[x,y]);
-- nd_weyl_gr :0.00335sec(0.003553sec)
-- weyl_minipoly_by_elim :0.00758sec(0.007857sec)
-- generic_bfct_and_gr :0.013sec(0.01995sec)
generic bfct : [[-1,1],[s,1],[s+a+b+c-1,1]]
S0 : 0
B_{S0} length : 1
-- fctr(BF) + base :0.001232sec(0.001368sec)
dimension : 1
[1]
```

パラメータは不定元として扱われるため, 整数でないという以外にも, パラメータが取ってはいけない値の集合 (除外集合) が存在する. 除外集合は, 積分アルゴリズムにおける b-関数の根でパラメータを含むものが非負整数にならないという条件と, Gröbner 基底計算の先頭項が消えないという条件から定まる. 除外集合を計算するには, オプション `excp` を用いる.

```
[1876] ns_twistedlog.difference_equation([x,y,1-x-y],[a,b,c],[x,y]|excp = 1);
Order : 1
[Operator,[ec*a^2+(ec-1)*c*a-ec*b^2+(-ec+1)*c*b,eb*a+(eb+ec-1)*b,ea+eb+ec-1],
Not integer,[a,b,c],Not non-negative integer,[-a-b-c+1],Not zero,[a,a-b,a+b+2
*c,a+b+c,b,a+c,a+b,c]]
```

この場合, 除外集合は

$$\{a \in \mathbb{Z}\} \cup \{b \in \mathbb{Z}\}\{c \in \mathbb{Z}\} \cup \{a+b+c-1 \in \mathbb{N} \cup \{0\}\} \cup \{a+b+c=0\} \cup \{a+b+2c=0\}$$
$$\cup \{a+b=0\} \cup \{a-b=0\} \cup \{a+c=0\} \cup \{a=0\} \cup \{b=0\} \cup \{c=0\}$$

である.

例 **5.9.** $p(a,b,c,d) = \int_C x^{a-1}y^{b-1}(1-x-y)^{c-1}(1-2x)^{d-1}dxdy$ とする. ただし $a,b,c,d,a+b+c+d \notin \mathbb{Z}, \mathrm{Re}(a), \mathrm{Re}(b), \mathrm{Re}(c), \mathrm{Re}(d) > 0$ とし, C は $x=0, y=0, 1-x-y=0, 1-2x=0$ で囲まれた $\mathbb{R}^2$ の有界領域の一つとする. アルゴリズム 5.1 より, p は次の差分方程式系を満たす:

$$\{(a+b+c+d)E_d^2 - 2(a-b-c)E_a + a - b - c - d\} \bullet p = 0,\ \{2E_a + E_d - 1\} \bullet p = 0,$$
$$\{2(b+c)(b+c+1)(a+b+c+d)E_b^2 + b(b+1)(a+3b+3c+2d)E_a$$
$$- b(b+1)(a+2b+2c+2d)\} \bullet p = 0, \{bE_a + (b+c)E_b - b\} \bullet p = 0,$$
$$\{2(b+c)(b+c+1)(a+b+c+d)E_c^2 + c(c+1)(a+3b+3c+2d)E_a$$
$$- c(c+1)(a+2b+2c+2d)\} \bullet p = 0,\ \{cE_a + (b+c)E_c - c\} \bullet p = 0,$$
$$\{2(a+b+c+d)E_a^2 - (3a+b+c+2d)E_a - a\} \bullet p = 0.$$

各変数に関して消去法を行うことにより, p は次の差分方程式系を満たす:

$$\{2(a+b+c+d)E_a^2 - (3a+b+c+2d)E_a + a\} \bullet p = 0,$$
$$\{2(b+c+1)(a+b+c+d)E_b^2 - (b+1)(a+3b+3c+2d)E_b + b(b+1)\} \bullet p = 0,$$
$$\{2(b+c+1)(a+b+c+d)E_c^2 - (c+1)(a+3b+3c+2d)E_c + c(c+1)\} \bullet p = 0,$$
$$\{(a+b+c+d)E_d^2 + (a-b-c)E_d - d\} \bullet p = 0.$$

コマンド ns_twistedlog.difference_equation で各変数に関する差分方程式を計算するにはオプション shift を用いる.

```
[1877]  ns_twistedlog.difference_equation([x,y,1-x-y,1-2*x],[a,b,c,d],[x,y]);
-- nd_weyl_gr :0.004001sec(0.001659sec)
-- weyl_minipoly_by_elim :0.004sec(0.005839sec)
Order : 2
[(2*ea-ed^2-1)*a+(-2*ea-ed^2+1)*b+(-2*ea-ed^2+1)*c+(-ed^2+1)*d,(-4*ea^2+3*ed^
2+1)*a+(4*ea^2+ed^2-1)*b+(4*ea^2+ed^2-1)*c+(2*ed^2-2)*d,((-2*eb-ed^2+1)*b-2*e
b*c)*a+(2*eb-ed^2-1)*b^2+((4*eb-ed^2-1)*c+(-ed^2+1)*d)*b+2*eb*c^2,((-4*eb^2-e
d^2+1)*b^2+(-8*eb^2*c-4*eb^2-ed^2+1)*b-4*eb^2*c^2-4*eb^2*c)*a+(4*eb^2-3*ed^2-
1)*b^3+((12*eb^2-3*ed^2-1)*c+(-2*ed^2+2)*d+4*eb^2-3*ed^2-1)*b^2+(12*eb^2*c^2+
(8*eb^2-3*ed^2-1)*c+(-2*ed^2+2)*d)*b+4*eb^2*c^3+4*eb^2*c^2,(2*ec*b+(2*ec+ed^2
-1)*c)*a-2*ec*b^2+(-4*ec+ed^2+1)*c*b+(-2*ec+ed^2+1)*c^2+(ed^2-1)*d*c,(-4*ec^2
*b^2+(-8*ec^2*c-4*ec^2)*b+(-4*ec^2-ed^2+1)*c^2+(-4*ec^2-ed^2+1)*c)*a+4*ec^2*b
^3+(12*ec^2*c+4*ec^2)*b^2+((12*ec^2-3*ed^2-1)*c^2+(8*ec^2-3*ed^2-1)*c)*b+(4*e
c^2-3*ed^2-1)*c^3+((-2*ed^2+2)*d+4*ec^2-3*ed^2-1)*c^2+(-2*ed^2+2)*d*c,(ed^2+e
d)*a+(ed^2-ed)*b+(ed^2-ed)*c+(ed^2-1)*d]
[1879]  ns_twistedlog.difference_equation([x,y,1-x-y,1-2*x],[a,b,c,d],[x,y]|s
hift = a);
-- nd_weyl_gr :0.006528sec(0.006691sec)
-- weyl_minipoly_by_elim :0.02177sec(0.02196sec)
Order : 2
[(-2*ea^2+3*ea-1)*a+(-2*ea^2+ea)*b+(-2*ea^2+ea)*c+(-2*ea^2+2*ea)*d]
[1880]  ns_twistedlog.difference_equation([x,y,1-x-y,1-2*x],[a,b,c,d],[x,y]|s
hift = b);
-- nd_weyl_gr :0.007233sec(0.007389sec)
-- weyl_minipoly_by_elim :0.02421sec(0.0244sec)
Order : 2
[((2*eb^2-eb)*b+2*eb^2*c+2*eb^2-eb)*a+(2*eb^2-3*eb+1)*b^2+((4*eb^2-3*eb)*c+(2
*eb^2-2*eb)*d+2*eb^2-3*eb+1)*b+2*eb^2*c^2+(2*eb^2*d+2*eb^2-3*eb)*c+(2*eb^2-2*
eb)*d]
[1881]  ns_twistedlog.difference_equation([x,y,1-x-y,1-2*x],[a,b,c,d],[x,y]|s
hift = c);
-- nd_weyl_gr :0.006166sec(0.006276sec)
-- weyl_minipoly_by_elim :0.02168sec(0.02187sec)
Order : 2
[(-2*ec^2*b+(-2*ec^2+ec)*c-2*ec^2+ec)*a-2*ec^2*b^2+((-4*ec^2+3*ec)*c-2*ec^2*d
-2*ec^2+3*ec)*b+(-2*ec^2+3*ec-1)*c^2+((-2*ec^2+2*ec)*d-2*ec^2+3*ec-1)*c+(-2*e
c^2+2*ec)*d]
[1882]  ns_twistedlog.difference_equation([x,y,1-x-y,1-2*x],[a,b,c,d],[x,y]|s
hift = d);
-- nd_weyl_gr :0.006479sec(0.006647sec)
-- weyl_minipoly_by_elim :0.02496sec(0.02515sec)
Order : 2
[(ed^2+ed)*a+(ed^2-ed)*b+(ed^2-ed)*c+(ed^2-1)*d]
```

5.2 $H^1(\Omega^\bullet(\log f),\nabla)$ の基底から定まる超幾何積分の場合

$f=\prod_{i=1}^s,\Phi=\prod_{i=1}^s f_i^{\alpha_i},\nabla=d+d\log\Phi$ とする. $g_1,\cdots,g_t\in\mathbb{C}[x,y]$ と f に付随する対数 1-形式 $\omega\in\Omega^1(\log f)$ に対して, 超幾何積分

$$q(\beta_1,\cdots,\beta_t)=\int_C\prod_{i=1}^t g_i^{\beta_i}\omega \tag{10}$$

を考える. (10) に対して次を仮定する:

$$\text{仮定:}g_i^\ell\omega\ (1\le i\le t,0\le\ell\le d)\text{ は全て }H^1(\Omega^\bullet(\log f),\nabla)\text{ に属する.} \tag{11}$$

ただし, $d=\dim H^1(\Omega^\bullet(\log f),\nabla)$ である. 積分路 C は (7) に対して述べた条件のうち, 2-単体の部分を 1-単体に変えたものをとる. この節では, 仮定 (11) を満たす超幾何積分 (10) の差分方程式系を求めるアルゴリズムを与える. アルゴリズムの実装は行っていない.

まず, 対数 1-形式 $\omega=\frac{pdx+qdy}{f}$ が $H^1(\Omega^\bullet(\log f),\nabla)$ の元となることと, ω が微分方程式

$$\partial_x\bullet q-\partial_y\bullet p+\frac{1}{f}(f_yq-f_xp)+\sum_{j=1}^s\frac{\alpha_j}{f_j}((f_j)_yq-(f_j)_xp)=0 \tag{12}$$

を満たすこととが同値であることに注意する.

$$E_{\beta_j}\bullet q=\int_C\prod_{i=1}^t g_i^{\beta_i}g_j\omega$$

が成り立つから, 仮定 (11) の下で, アルゴリズム 5.1 において $H^2(\Omega^\bullet(\log f),\nabla)$ の基底 $c_1,\cdots,c_m$ を $H^1(\Omega^\bullet(\log f),\nabla)$ の基底 $\omega_1,\cdots,\omega_d$ に, f_i^ℓ を $g_i^\ell\omega$ に置き換えることで, $q(\beta)$ の満たす差分方程式系を計算するアルゴリズムが得られる. 斉次の場合には定理 5.3 と同様にして, 差分方程式系は 0 次元であることが証明できる.

例 **5.10.**

$$q(k,m)=\int_C(x^ky^{n-k+1}+x^{n-m+1}y^m)\frac{ydx-xdy}{xy(x-y)}$$

を考える. ただし, k,m,n は 0 以上の整数であり $0\le k,\ m\le n+1$ であるとする. また, C は注意 5.4 の条件で対数 1-形式の部分を対数 0-形式に, Φ の部分を $x^ky^{n-k+1}+x^{n-m+1}y^m$ に変えた条件を満たすと仮定する (この例の場合の Φ の取り方は後述). $f=xy(x-y)$ とすると, $\Omega^1(\log f)=\mathbb{C}[x,y]\omega_1\bigoplus\mathbb{C}[x,y]\omega_2$ となる. ここで, $\omega_1=\frac{ydx-xdy}{f},\omega_2=\frac{xy(x-y)}{f}=\frac{1}{2}d\log f$ である. $\Phi=(x-y)^{-n}$ のとき, $h_1\omega_1+h_2\omega_2\in H^1(\Omega^\bullet(\log f),\nabla)$ に対して, h_1,h_2 は次の微分方程式の解となる:

$$(x\partial_x+y\partial_y)\bullet h_1-(1+n)h_1+xy(\partial_x+\partial_y)\bullet h_2=0$$

これより特に, $x^k y^{n-k+1}\omega_1$ $(0 \leq k \leq n+1)$ は $H^1(\Omega^\bullet(\log f), \nabla)$ の元であることが分かる. ところで, $x^k y^{n-k} \in \Omega^0(\log f)$ に対して,

$$\nabla(x^k y^{n-k}) = (kx^k y^{n-k+1} + (n-k)x^{k+1}y^{n-k})\omega_1$$

となるから,

$$kx^k y^{n-k+1}\omega_1 \equiv -(n-k)x^{k+1}y^{n-k}\omega_1 \ (\mathrm{mod}\nabla) \tag{13}$$

が成り立つ.

$E_k \bullet q = \int_C (x^{k+1}y^{n-k} + x^{n-m+1}y^m)\omega_1$, $E_m \bullet q = \int_C (x^k y^{n-k+1} + x^{n-m}y^{m+1})\omega_1$

であるから, (13) より

$$E_k \bullet q = \frac{k}{k-n}q, \ E_m \bullet q = \frac{m-n}{m}q,$$

となる.

6 指数関数と多項式ベキの積分から定まるある超幾何積分が満たす微分方程式系への応用

6.1 $H^2(\Omega^\bullet(\log f), \nabla)$ の基底から定まる超幾何積分の場合

$f_1, \cdots, f_s \in \mathbb{C}[x, y]$, $g(t, x, y) \in \mathbb{C}[t_1, \cdots, t_m, x, y], \alpha_1, \cdots, \alpha_s \in \mathbb{C}$ に対して

$$F(t_1, \cdots, t_m) = \int_C \exp(g(t, x, y)) \prod_{i=1}^{s} f_i(x, y)^{\alpha_i - 1} dxdy \tag{14}$$

とおく. $\Phi = \exp(g(t_1, \cdots, t_m, x, y)) \prod_{i=1}^s f_i(x, y)^{\alpha_i}, f = \prod_{i=1}^s f_i(x, y)$ とおくと $F(t) = \int_C \Phi \frac{dxdy}{f}$ と書ける. 積分路 C は (7) に対して述べた条件のうち, 差分の部分を微分に変えたものをとる. この節では超幾何積分 (14) が満たす微分方程式系を求めるアルゴリズムを与える. このアルゴリズムの実装は数式処理システム Risa/Asir で行った. ただし, 実装したプログラムは $\alpha_1, \cdots, \alpha_s \notin \mathbb{Z}$ である場合にのみ対応している. プログラムはパッケージ `ns_twistedlog.rr` に収められている.

このとき,

$$\partial_{t_i}^\ell \bullet F(t) = \int_C \Phi \frac{1}{\exp(g)} \partial_{t_i}^\ell \bullet \exp(g) \frac{dxdy}{f}$$

となることに注意すると, アルゴリズム 5.1 において (5) を $L_i + q_i g_x - p_i g_y$ に, f_i^ℓ を $\frac{1}{\exp(g)}\partial_{t_i}^\ell \bullet \exp(g)$ に置き換えることで, $F(t)$ の満たす非斉次微分方程式系を計算するアルゴリズムが得られる. さらに定理 5.3 の証明と同様にして, 斉次の場合にはこのアルゴリズムから得られる微分方程式系は 0 次元であることが証明できる.

例 **6.1.**

$$F(t_1,t_2)=\int_C \exp(t_1x^2+t_2y^2)(x+y)^{a-1}dxdy$$

を考える. ただし $a\notin\mathbb{Z}$ とし, C は注意 5.4 の条件 1 を満たすものをとる. $H^2(\Omega^\bullet(\log f),\nabla)=2$ であり, $F(t_1,t_2)$ が満たす微分方程式系は次のようになる.

$$\{2t_1(t_1+t_2)\partial_{t_1}+t_1+at_2\}\bullet F=0,\ \{2t_2(t_1+t_2)\partial_{t_2}+t_2+at_1\}\bullet F=0,$$
$$\{4t_1(t_1+t_2)\partial_{t_1}^2-3t_1^2-6at_1t_2-a(a+2)t_2^2\}\bullet F=0,$$
$$\{4t_2(t_1+t_2)\partial_{t_2}^2-3t_2^2-6at_1t_2-a(a+2)t_1^2\}\bullet F=0$$

この微分方程式系はホロノミックになっており, ホロノミックランクは 1 である.

Risa/Asir で超幾何積分 (14) の満たす微分方程式系を計算するコマンドは `ns_twistedlog.differential_equation` である. また, D の左イデアルがホロノミックかどうかの判定はコマンド `ns_twistedlog.holonomic` で行える.

```
[1883] ns_twistedlog.differential_equation(t1*x^2+t2*y^2,[x+y],[a],[x,y],[t1,t2]);
-- nd_weyl_gr :0.004739sec(0.004889sec)
-- weyl_minipoly_by_elim :0.00437sec(0.004545sec)
Order : 2
[(2*t1*t2+2*t1^2)*dt1+a*t2+t1,(-4*t1^2*t2^2-8*t1^3*t2-4*t1^4)*dt1^2+(a^2+2*a)*t2^2+
6*a*t1*t2+3*t1^2,(2*t2^2+2*t1*t2)*dt2+t2+a*t1,(-4*t2^4-8*t1*t2^3-4*t1^2*t2^2)*dt2^2
+3*t2^2+6*a*t1*t2+(a^2+2*a)*t1^2]
[1884] G = @@$
[1885] ns_twistedlog.holonomic(G,[t1,t2],[dt1,dt2]);
Hilbert polynomial : 7/2*x^2-1/2*x+2
holonomic : Yes
holonomic rank : 1
```

6.2 $H^1(\Omega^\bullet(\log f),\nabla)$ の基底から定まる超幾何積分の場合

$f=\prod_{i=1}^s f_i(x,y),\Phi=\prod_{i=1}^s f_i^{\alpha_i},\nabla=d+d\log\Phi\wedge$ とする. f に付随する対数 1-形式 $\omega=\frac{pdx+qdy}{f}$ に対して, 超幾何積分

$$G(t_1,\cdots,t_m)=\int_C \exp(g(t_1,\cdots,t_m,x,y))\prod_{i=1}^t h_i(x,y)^{\beta_i}\omega \qquad (15)$$

を考える. (15) に対して, 次の仮定を置く:

$$\text{仮定:}\frac{1}{\exp(g)}\partial_{t_i}^\ell\bullet\exp(g)\omega\ (1\le i\le m,0\le\ell\le d)\text{が}H^1(\Omega^\bullet(\log f),\nabla)\text{に属する.} \qquad (16)$$

ただし, $d=\dim H^1(\Omega^\bullet(\log f),\nabla)$ と置いた. 積分路 C は (7) に対して述べた条件のうち, 2-単体の部分を 1-単体に置き換えたものをとる. この節では,

仮定 (16) を満たす超幾何積分 (15) の微分方程式系を計算するアルゴリズムを与える. アルゴリズムの実装は行っていない.

$G(t)$ が仮定 (16) を満たすことは, $\frac{1}{\exp(g)}\partial_{t_i}^{\ell} \bullet \exp(g)\omega$ が微分方程式 (12) を満たすことと同値である.

$$\partial_{t_i}^{\ell} \bullet G(x) = \prod_{i=1}^{t} h_i^{\beta_i} (\frac{1}{\exp(g)}\partial_{t_i}^{\ell} \bullet \exp(g))\omega$$

が成り立つから, 仮定 (16) の下で 5.1 において $H^2(\Omega^\bullet(\log f), \nabla)$ の基底 $c_1, \cdots, c_m$ を $H^1(\Omega^\bullet(\log f), \nabla)$ の基底 $\omega_1, \cdots, \omega_d$ に, f_i^{ℓ} を $\frac{1}{\exp(g)}\partial_{t_i}^{\ell} \bullet \exp(g)\omega$ に置き換えることで, $G(x)$ の満たす微分方程式系を計算するアルゴリズムが得られる. 斉次の場合には定理 5.3 と同様にして, 微分方程式系は 0 次元であることが証明できる.

例 **6.2.**

$$G(t_1, t_2) = \int_C \exp(\frac{x}{y}t_1 + \frac{y}{x}t_2)x^k y^{n-k+1}\omega$$

を考える. ここで, $\omega = \frac{ydx - xdy}{xy(x-y)}$ である. また, k, n は 0 以上の整数で $0 \leq k \leq n+1$ とし, C は注意 5.4 の条件で対数 1-形式の部分を対数 0-形式に, Φ の部分を $x^k y^{n-k+1}$ に置き換えた条件を満たすと仮定する (この例の場合の Φ の取り方は後述). $f = xy(x-y), \Phi = (x-y)^{-n}$ とおくと, 例 5.10 と同様にして, $x^k y^{n-k+1}\omega$ $(0 \leq k \leq n+1)$ は $H^1(\Omega^\bullet(\log f), \nabla)$ の元であり, これらの間に (13) と同様の関係式が成り立つ.

$$\partial_{t_1} \bullet G = \int_C \exp(\frac{x}{y}t_1 + \frac{y}{x}t_2)x^{k+1}y^{n-k}\omega,\ \partial_{t_2} \bullet G = \int_C \exp(\frac{x}{y}t_1 + \frac{y}{x}t_2)x1^{k-1}y^{n-k+2}\omega$$

となるから,

$$\partial_{t_1} \bullet G = \frac{k}{k-n}G,\ \partial_{t_2} \bullet G = \frac{k-n-1}{k-1}G$$

が成り立つ.

7 アルゴリズムの比較・検討

(7), (14) の形の超幾何 2 重積分の満たす差分方程式系, 微分方程式系は Chyzak アルゴリズム [5, 6, 7] でも計算することができる. Chyzak アルゴリズムは Maple の `Mgfun` パッケージ [29] のコマンド `creative_telescoping` に実装されている. ただし, `creative-telescoping` は (10), (15) の形の超幾何 1 重積分の満たす差分方程式系, 微分方程式系の計算には対応していない.

例 **7.1.** Maple14 上で `creative_telescoping` を用いて $p(a,b,c) = \int_C x^{a-1}y^{b-1}(1-x-y)^{c-1}dxdy$ の満たす差分方程式系を計算すると次のように出力される.

```
> with(Mgfun);
> f := x^(a-1)*y^(b-1)*(1-x-y)^(c-1);
f := x^(a-1)*y^(b-1)*(1-x-y)^(c-1)
> l := creative_telescoping(f,[a::shift,b::shift,c::shift,x::diff],y::diff);
l := [[-x*_F(a, b, c, x)+_F(a+1, b, c, x), 0], [(-b+b*x)*_F(a, b, c, x)+(b+c)*_
F(a, b+1, c, x), (-y+y*x+y^2)*_f(a, b, c, x, y)], [(c*x-c)*_F(a, b, c, x)+(b+c)
*_F(a, b, c+1, x), (y-y*x-y^2)*_f(a, b, c, x, y)], [(-1+2*x+a-a*x-b*x-c*x)*_F(a
, b, c, x)+(-x+x^2)*(diff(_F(a, b, c, x), x)), -y*x*_f(a, b, c, x, y)]]
> sys := {l[1][1],l[2][1],l[3][1],l[4][1]};
sys := {-x*_F(a, b, c, x)+_F(a+1, b, c, x), (-b+b*x)*_F(a, b, c, x)+(b+c)*_F(a,
 b+1, c, x), (c*x-c)*_F(a, b, c, x)+(b+c)*_F(a, b, c+1, x), (-1+2*x+a-a*x-b*x-c
*x)*_F(a, b, c, x)+(-x+x^2)*(diff(_F(a, b, c, x), x))}
> creative_telescoping(LFSol(sys),[a::shift,b::shift,c::shift],x::diff);
[[-a*_F(a, b, c)+(b+a+c)*_F(a+1, b, c), (-x+x^2)*_f(a, b, c, x)], [-b*_F(a, b,
c)+(b+a+c)*_F(a, b+1, c), (b*x-x^2*b)*_f(a, b, c, x)/(b+c)], [-c*_F(a, b, c)+(b
+a+c)*_F(a, b, c+1), (c*x-c*x^2)*_f(a, b, c, x)/(b+c)]]
```

表 5 はアルゴリズム 5.1 と Chyzak アルゴリズムの計算時間を斉次差分方程式系の場合に比較したものである. 計算時間の比較は Intel Xeon E5410(2.33GHz) で行った. 項目 System は差分方程式系を計算するのに費やした時間, 項目 a_1-shift は差分作用素を E_{a_1} しか含まない差分方程式を計算するのに費やした時間である. ただしアルゴリズム 5.1 の a_1-shift における計算は, 注意 5.5 に述べた方法を用いている. また Chyzak アルゴリズムでは, 差分作用素系の計算時間 (項目 System の時間) + 消去法にかかった時間を a_1-shift での経過時間としている.

表 5: The comparison of the computing time (seconds)

Integrand	Alg. 5.1				Chyzak alg.		
	$\mathrm{Dim}H^2$	Rank	System	a_1-shift	Rank	System	a_1-shift
F_1	2	2	0.04	0.024	2	1.35	1.35
F_2	3	3	0.28	0.096	3	12.67	12.75
F_3	4	4	96.24	10.00	4	34.30	35.53
F_4	5	—	30.49	161.19	—	—	—
$G_{3,5}$	2	2	0.27	0.27	2	1.61	1.61
$G_{3,6}$	3	3	35.5	35.5	1	5.72	5.72
$G_{3,7}$	—	—	>3 hours	>3 hours	4	10.25	10.25
$H_{3,4}$	2	2	4.56	8.44	1	37.87	37.87
$H_{3,5}$	3	3	490.25	457.20	—	—	—
$H_{3,6}$	—	—	>3 hours	>3 hours	3	107.83	108.64

$F_1 = F(1-2x)^{a_4-1},\ F_2 = F(1-2x-3y)^{a_4-1},\ F_3 = F(1-2x-3y)^{a_4-1}(1-x)^{a_5-1},$
$F_4 = F(1-2x-3y)^{a_4-1}(1-2y)^{a_5-1},\ G_{p,q} = (x^p+y^q+xy^{q-1})^{a_1-1},\ H_{p,q} = G_{p,q}(x+y)^{a_2-1}$

ただし, $F = x^{a_1-1}y^{a_2-1}(1-x-y)^{a_3-1}$

表 5 から分かるように, アルゴリズム 5.1 は $H^2(\Omega(\log f), \nabla)$ の次元が大きいほど, また因子の次数が大きくなるほど計算時間が長くなる傾向にある. また, 差分方程式系を計算するよりも E_{a_1} に関する差分方程式を計算する方が計算時間が短い場合がある.

F_1, F_2, F_3, F_4 の場合には, F_3 を除けばアルゴリズム 5.1 の方が計算時間が短い. $G_{p,q},\ H_{p,q}$ の場合には, (p,q) が小さい場合を除いて Chyzak アルゴリズムの方が計算時間が短い.

$G_{3,7}, H_{3,6}$ の場合にボトルネックになっているのは, 積分アルゴリズムにおけるパラメータ付きの b-関数の計算である. これは, 積分アルゴリズムの計算に使用しているパッケージ `nk_restriction.rr` では, パラメータ付きの b-関数の計算が野呂 [17] による効率のよいアルゴリズムではなく, 消去法によってなされるためである. いくつか計算を行った結果, 多項式の次数が大きい場合や, 多項式に含まれる単項式の次数の差が大きい場合には, 消去法によるパラメータ付きの b-関数の計算時間が長くなる傾向があることが分かった.

以上のことから, 多項式の次数が大きくない場合や, 多項式に含まれる単項式の次数の差が大きくない場合には twisted logarithmic cohomology 群の方法が, それ以外の場合には Chyzak の方法が効率がよいと考えられる.

3 変数以上の場合, (5) を 3 変数以上の場合に拡張した微分作用素がなす

右イデアルがホロノミックの場合には, twisted logarithmic cohomology 群の方法が適用できる. その場合には Chyzak の方法より効率がよい場合があるが, twisted logarithmic cohomology 群の方法では計算できない場合があるので, 注意が必要である. 実際, [10, 6.4 Example4] の多項式に対しては twisted logarithmic cohomology 群の方法は適用できない. また表には挙げていないが, 積分アルゴリズムの方法は, 表で用いた例に対しては twisted logarithmic cohomology 群の方法, Chyzak の方法よりも格段に遅くなってしまう. ただし, 例えば $F(x,y) = \int_a^b \frac{1}{xt+y+t^{10}} dt$ のような "疎" な多項式の積分の満たす微分方程式系の計算に対しては, Chyzak アルゴリズムよりも積分アルゴリズムの方法が効率のよいことが知られている [16, Example 3]. $F(x,y)$ の満たす微分方程式系は twisted logarithmic cohomology 群の方法では求められない.

参考文献

[1] G. Almkvist, D. Zeilberger, The method of differentiating under the integral sign, Journal of Symbolic Computation **10**, 571-591, 1990.

[2] 青本和彦, 喜多通武, 超幾何関数論, シュプリンガー・フェアラーク東京株式会社, 1994.

[3] J. Björk, *Rings of Differential Operators*, North-Holland, New York, 1979.

[4] F.J. Calderón-Moreno, Logarithmic differential operators and logarithmic de Rham complexes relative to a free divisor, Annales Scientifiques de lÉcole Normale Supérieure (4) **32**, 701-714, 1999.

[5] F. Chyzak, Gröbner Bases, Symbolic Summation and Symbolic Integration, London Mathematics Lecture Notes Series, val.251, 32-60, 1998.

[6] F. Chyzak, An Extension of Zeilbeger's Fast Algorithm of General Holonomic Functions, Discrete Mathematics **217**, 115-134, 2000.

[7] F. Chyzak and B. Salvy, Non-commutative elimination in Ore algebras proves multivariate identities, *J. Symbolic Computation*, vol. 26, 187-227, 1998.

[8] F.J. Castro-Jiménez and N. Takayama, The computation of the logarithmic cohomology for plane curves, Journal of Algebra. **322**, 3839-3851, 2009.

[9] F.J. Castro-Jiménez and J.M. Ucha, Explicit computation theorems for D-modules, Journal of Symbolic Computation **32**, 677-685, 2001.

[10] F.J. Castro-Jiménez and J.M. Ucha, Gröbner bases and logarithmic $\mathcal{D}$-modules, Journal of Symbolic Computation, **41**, 317-335, 2006.

[11] D. Cox, J. Little, D. O'Shea, *Using Algebraic Geometry*, Springer, New York, 1998.

[12] D. Eisenbud, *Commutative Algebra with a view toward Algebraic Geometry*, Springer, New York, 1995.

[13] A. Logar, B. Sturmfels, Algorithms for the Quillen-Suslin Theorem, Journal of Algebra, **145**, 231-239, 1992.

[14] P.J. Hilton, U. Stammbach, *A course in homological algebra*, Springer, 1970.

[15] 松村英之, 可換環論, 共立出版, 1980.

[16] H. Nakayama, K. Nishiyama, An algorithm of computing inhomogeneous differential equations for definite integrals, Lecture Notes In Computer Science, **6327**, 221-232, 2010.

[17] M. Noro, An efficient modular algorithm for computing the global B-function, in: A.M. Cohen, X.S. Gao, N. Takayama (Eds.), Mathematical Software, Proceedings of the First International Congress of Mathematical Software, World Scientific, Beijing, pp. 147-157, 2002.

[18] T. Oaku, Algorithms for b-functions, restrictions, and algebraic local cohomology groups of D-modules, Advances in Applied Mathematics, **19**, 61-105, 1997.

[19] 大阿久俊則, D 加群と計算数学, 朝倉書店, 2002.

[20] T. Oaku Y. Shiraki and N. Takayama, Algebraic Algorithm for D-modules and numerical analysis, Computer mathematics (Proceedings of ASCM 2003), 23-39, Lecture Notes Ser. Comput., 10, World Sci. Publ., River Edge, NJ, 2003.

[21] T. Oaku and N. Takayama, An algorithm for de Rham cohomology groups of the complement of an affine variety via D-module computation, Journal of Pure and Applied Algebra, **139**, 201-233, 1999.

[22] T. Oaku and N. Takayama, Algorithms for D-modules−restriction, tenser product, localization, and local cohomology groups, Journal of Pure and Applied Algebra **156**, 267-308, 2001.

[23] T. Oaku, N. Takayama, H. Tsai, Polynomial and Rational Solutions of Holonomic Systems, Journal of Pure and Applied Algebra, **164**, 199 - 220, 2001.

[24] O. Ore, Theory of non-commutative polynomials, Annals of Mathematics, **34**, 480-508, 1933.

[25] K. Saito, Theory of logarithmic differential forms and logarithmic vector fields, Journal of Faculty of Science, University of Tokyo. Section IA. **27**, 265-291, 1980.

[26] M. Saito, B. Sturmfels, N. Takayama, *Gröbner deformations of hypergeometric differential equations*, Algorithms and Computation in Mathematics, 6. Springer-Verlag, Berlin, 2000. viii+254 pp.

[27] D. Zeilberger, A holonomic systems approach to special functions identities, Journal of Computational and Applied Mathematics **32**, 321-368, 1990.

[28] D. Zeilberger, The method of creative telescoping, Journal of Symbolic Computation **11**, 195-204, 1991.

[29] F. Chyzak, `Mgfun`,
`http://algo.inria.fr/chyzak/mgfun.html`

[30] D. Grayson and M. Stillman, Macaulay2, a software system for research in algebraic geometry,
`http://www.math.unic.edu/macaulay2`

[31] A. Leykin, H, Tsai, D-module package for Macaulay2,
`http://www.math.uiuc.edu/Macaulay2`

[32] H. Nakayama, K. Nishiyama, `nk_restriction,rr`,
`http://www.math.kobe-u.ac.jp/~nakayama/nk_restriction.rr`

[33] K. Nishitani, `twistedLogCohomology.m2`,
`http://www.math.sci.kobe-u.ac.jp/cgi/cvsweb.cgi/OpenXM/src/math-misc/twistedLogCohomology.m2`

[34] K. Nishitani, `ns_twistedlog.rr`,
`http://www.math.sci.kobe-u.ac.jp/cgi/cvsweb.cgi/OpenXM/src/asir-contrib/packages/src/ns_twistedlog.rr`

[35] M. Noro, et al., Risa/Asir,
`http://www.math.kobe-u.ac.jp/Asir/asir-jp.html`