数理解析研究所講究録
第1815巻 2012年 112-123

積分の満たす非斉次微分方程式系を与えるアルゴリズム
An algorithm of computing inhomogeneous differential equations for definite integrals

中山 洋将
NAKAYAMA HIROMASA
神戸大学大学院理学研究科 / JST CREST
DEPARTMENT OF MATHEMATICS, GRADUATE SCHOOL OF SCIENCE, KOBE UNIVERSITY *

西山 絢太
NISHIYAMA KENTA
神戸大学大学院理学研究科 / JST CREST
DEPARTMENT OF MATHEMATICS, GRADUATE SCHOOL OF SCIENCE, KOBE UNIVERSITY †

1 イントロダクション

n 変数 $x_1,\ldots,x_n$ に対する微分作用素を $\partial_1,\ldots,\partial_n$ として, 微分作用素環 D とその部分環 D' を

$$D=K\langle x_1,\ldots,x_n,\partial_1,\ldots,\partial_n\rangle,\quad D'=K\langle x_{m+1},\ldots,x_n,\partial_{m+1},\ldots,\partial_n\rangle$$

とおく. ここで, $m\leq n$ とし, 係数体 K は有理数体としておく. このとき, ホロノミックな左 D イデアル I の $x_1,\ldots,x_m$ についての積分イデアル J とは

$$J=(I+\partial_1 D+\cdots+\partial_m D)\cap D'$$

なる左 D' イデアルのことであり, 積分イデアルを計算するアルゴリズムは大阿久 [6] により与えられた. 被積分関数の満たす微分方程式系が I の時, I の $x_1,\ldots,x_m$ についての積分イデアル J は, 積分路がサイクルである, パラメータ $x_{m+1},\ldots,x_n$ を含む積分の満たす斉次微分方程式系を与える.

我々はこの積分アルゴリズムを改良することにより, 積分イデアル J の生成元を求めるだけでなく, 各生成元 $P\in J$ について,

$$P=P_0+\partial_1 P_1+\cdots+\partial_m P_m$$

となる $P_0\in I, P_1,\ldots,P_m\in D$ を計算するアルゴリズムを与えた. このような $P_1,\ldots,P_m$ を P の非斉次部分と呼ぶことにする. 非斉次部分を計算することにより, パラメータを含む積分の満たす非斉次微分方程式系を得られる.

*nakayama@math.kobe-u.ac.jp
†nisiyama@math.kobe-u.ac.jp

例えば, $m=1, n=2$ として (すなわち, $D=K\langle x_1,x_2,\partial_1,\partial_2\rangle, D'=K\langle x_2,\partial_2\rangle$), 積分 $A(x_2)=\int_a^b e^{-x_1-x_2x_1^3}dx_1$ の満たす微分方程式を考える. 被積分関数 $f(x_1,x_2)=e^{-x_1-x_2x_1^3}$ の D における零化イデアル $I=\langle\partial_1+1+3x_2x_1^2, \partial_2+x_1^3\rangle$ の x_1 に関する積分イデアルは $J=\langle 27x_2^3\partial_2^2+54x_2^2\partial_2+6x_2+1\rangle=\langle P\rangle$ である. J の生成元 P の非斉次部分として $P_1=-(\partial_1^2+3\partial_1+3)$ が得られる. 今,

$$P=P_0+\partial_1 P_1 \quad (P_0\in I)$$

が成り立つ. この P を積分に作用させることで

$$\begin{aligned} P\cdot A(x) &= \int_a^b \partial_1(P_1\cdot e^{-x_1-x_2x_1^3})dx_1 = \left[P_1\cdot e^{-x_1-x_2x_1^3}\right]_{t=a}^{t=b} \\ &= -\left[(9x_2^2x_1^4-3x_2x_1^2-6x_2x_1+1)e^{-x_1-x_2x_1^3}\right]_{t=a}^{t=b} \end{aligned}$$

が得られ, 積分 $A(x)$ の満たす非斉次微分方程式が得られる. 今までの積分アルゴリズムでは, 積分イデアルの生成元だけしか得られず, 非斉次部分を計算できないので, 上のような計算はできなかった.

この論文では, 積分イデアルの非斉次部分の計算アルゴリズムとその適用例について述べる. また, 積分の満たす非斉次微分方程式系を計算する他のアルゴリズムとして, Almkvist-Zeilberger アルゴリズム ([1]) や Chyzak アルゴリズム ([3],[4]), Oaku-Shiraki-Takayama アルゴリズム ([7]) などがある. それらアルゴリズムと我々のアルゴリズムとの比較についても述べる. 更に, 我々はこの論文で述べたアルゴリズムを, 数式処理ソフト Risa/Asir ([11]) において実装し, パッケージ `nk_restriction.rr` ([14]) として公開している.

2 D 加群の積分アルゴリズム

この節では, D 加群における積分アルゴリズムを解説する.

微分作用素環 $D=K\langle x_1,\ldots,x_m,x_{m+1},\ldots x_n,\partial_1,\ldots,\partial_m,\partial_{m+1},\ldots,\partial_n\rangle$, とその部分環 $D'=K\langle x_{m+1},\ldots,x_n,\partial_{m+1},\ldots,\partial_n\rangle$ を定めておく. (ただし, $m\leq n$)

D の環同型写像 $\mathcal{F}$ を

$$\mathcal{F}(x_i)=\begin{cases}-\partial_i & (1\leq i\leq m)\\ x_i & (m+1\leq i\leq n)\end{cases}, \mathcal{F}(\partial_i)=\begin{cases}x_i & (1\leq i\leq m)\\ \partial_i & (m+1\leq i\leq n)\end{cases}$$

なるものとし, Fourier 変換と呼ぶ.

ホロノミック左 D イデアル I の $x_1,\cdots,x_m$ についての積分イデアル J とは

$$J=(I+\partial_1 D+\cdots+\partial_m D)\cap D'$$

なる左 D' イデアルである.

アルゴリズム 1 (D 加群の積分アルゴリズム, [6], [8])

入力 : ホロノミック左 D イデアル I の生成元.

重みベクトル $w=(w_1,\ldots,w_m,w_{m+1},\ldots,w_n)$ で $w_1,\ldots,w_m>0, w_{m+1}=\cdots=w_n=0$ を満たすもの.

出力 : I の $x_1,\ldots,x_m$ についての積分イデアルの生成元.

1. $\mathcal{F}(I)$ に対し w についての制限加群の計算. 具体的には次のように行う.

   (a) 左 D イデアル $\mathcal{F}(I)$ の単項式順序 $<_{(-w,w)}$ についてのグレブナ基底 $\{h_1,\ldots,h_l\}$ を計算.

(b) 左 D イデアル $\mathcal{F}(I)$ の重みベクトル $(-w,w)$ についての generic b 関数 $b(s)$ を計算.
s_0 を $b(s)$ の非負最大整数根とする.

(c) s_0 がない時, 制限加群は 0 であり, 積分イデアルは D' となり, アルゴリズムを終了.

(d) $m_i = \mathrm{ord}_{(-w,w)}(h_i)$,
$\mathcal{B}_d = \{\partial_1^{i_1}\cdots\partial_m^{i_m} \mid i_1,\ldots,i_m \in \mathbb{Z}_{\geq 0}, i_1w_1+\cdots+i_mw_m \leq d\}$ $(d \in \mathbb{Z}_{\geq 0})$,
$r = \#\{(i_1,\ldots,i_m) \mid i_1,\ldots,i_m \in \mathbb{Z}_{\geq 0}, i_1w_1+\cdots+i_mw_m \leq s_0\}$ とおく.

(e) $\tilde{\mathcal{B}} = \bigcup_{i=1}^{l}\{\tilde{h}_{i\beta} := \partial^\beta h_i \mid \partial^\beta \in \mathcal{B}_{s_0-m_i}\}$ とし,
$\mathcal{B} = \{h_{i\beta} := \tilde{h}_{i\beta}|_{x_1=\cdots=x_m=0} \mid \tilde{h}_{i\beta} \in \tilde{\mathcal{B}}\}$ を返す.
ここで, $\tilde{h}_{i\beta}|_{x_1=\cdots=x_m=0}$ は, $\tilde{h}_{i\beta}$ を微分作用素の標準的な表示 $\sum c_{u,v}x^u\partial^v$ の形に直してから $x_1=0,\ldots,x_m=0$ を代入する操作を表す.

2. $\mathcal{F}^{-1}(\mathcal{B})$ を $(D')^r$ の元たちとみなし, それらの生成する左 D' 加群を M とおく.
ここで具体的な対応について説明する. $\mathcal{F}^{-1}(\mathcal{B})$ の各元の $\partial_1,\ldots,\partial_m$ の指数は 0 であり, $(-w,w)$ についての階数が s_0 以下であることから,

$$\sum_{i_1w_1+\cdots+i_mw_m\leq s_0} c_{i_1,\ldots,i_m}x_1^{i_1}\cdots x_m^{i_m} \quad (c_{i_1,\ldots,i_m} \in D')$$

の形に書ける. これを $(D')^r$ の元 $(c_{i_1,\ldots,i_m})_{i_1,\ldots,i_m}$ とみなす.

3. 左 D' 加群 M に対し, x^0 に対応する位置が最小になるような POT (Position Over Term) 項順序について, グレブナ基底を計算する. その生成元の中で, x^0 に対応する成分以外が全て 0 になっているものを取り出し, その x^0 成分たちからなる集合が I の積分イデアルの生成系になる.

I を被積分関数の満たす微分方程式系としたとき, その積分イデアルは積分の満たす斉次微分方程式系に対応している. まず, ホロノミックな関数 $f(x_1,\ldots,x_n)$ の $x_1,\ldots,x_m$ に関する定積分

$$A(x_{m+1},\ldots,x_n) = \int_R f(x_1,\ldots,x_n)dx_1\cdots dx_m, \quad R = \prod_{i=1}^m [a_i,b_i]$$

を考える. f の D における零化イデアル $I = \mathrm{Ann}_D f$ の $x_1,\ldots,x_m$ に関する積分イデアルを J とすると, 任意の $p \in J$ に対して,

$$p - \sum_{i=1}^m \partial_i p_i \in I$$

となるような $p_1,\ldots,p_m \in D$ が存在する. この微分作用素 p を積分 $A(x_{m+1},\ldots,x_n)$ に作用させると, 微分方程式

$$p\cdot A(x_{m+1},\ldots,x_n) = \int_R p\cdot f dt = \int_R \sum_{i=1}^m (\partial_i p_i)\cdot f dt = \sum_{i=1}^m \int_R \partial_i(p_i\cdot f)dt \tag{1}$$

が得られる. つまり, 積分イデアルの元たち p は, $A(x_1,\ldots,x_m)$ の積分領域として式 (1) の最右辺の積分たちが 0 となるようなものを取ったとき, $p\cdot A(x_{m+1},\ldots,x_n) = 0$ という斉次微分方程式を満たす.

3 積分の満たす非斉次微分方程式系を与えるアルゴリズム

この節では, 前節の積分アルゴリズムを改良して, 積分イデアルの非斉次部分の計算アルゴリズムを与える. また, このアルゴリズムの応用として, 積分が満たす非斉次な微分方程式系を得られることを述べる.

前節の最後の議論から, 式 (1) の右辺が 0 とならないような積分領域の場合は, p_i $(1 \leq i \leq m)$ を具体的に求める必要がある.

定理 1
ホロノミックな 左 D イデアル I の積分イデアルを $J \subset D'$ とする. このとき, 任意の $p \in J$ に対して

$$p - \sum_{i=1}^{m} \partial_i p_i \in I \tag{2}$$

を満たす微分作用素 $p_i \in D$ $(1 \leq i \leq m)$ をアルゴリズミックに計算できる.

証明 アルゴリズム 1 を適用して積分イデアル J の生成系 $\{g_1, \ldots, g_t\}$ を得る. この各生成元 g_j $(1 \leq j \leq t)$ に対して示せば十分である. アルゴリズム 1 の 3. より, $q_{ji\beta} \in D'$ を用いて $g_j = \sum_{i,\beta} q_{ji\beta} \mathcal{F}^{-1}(h_{i\beta})$ と表現される. また, この $q_{ji\beta}$ はグレブナ基底計算における履歴を参照することで計算できる. また, $\mathcal{F}^{-1}(\tilde{h}_{i\beta}) = \mathcal{F}^{-1}(\partial^\beta h_i) = \mathcal{F}^{-1}(\partial^\beta)\mathcal{F}^{-1}(h_i)$ であり, $\mathcal{F}(I)$ の生成系 (特にグレブナ基底) が $\{h_1, \ldots, h_l\}$ であったから, $\{\mathcal{F}^{-1}(h_1), \ldots, \mathcal{F}^{-1}(h_l)\}$ は I の生成系となり, $\mathcal{F}^{-1}(\tilde{h}_{i\beta}) \in I$ である. 従って,

$$\begin{aligned}
I \ni \sum_{i,\beta} q_{ji\beta} \mathcal{F}^{-1}(\tilde{h}_{i\beta}) &= g_j - \left(g_j - \sum_{i,\beta} q_{ji\beta} \mathcal{F}^{-1}(\tilde{h}_{i\beta}) \right) \\
&= g_j - \sum_{i,\beta} q_{ji\beta} \left(\mathcal{F}^{-1}(h_{i\beta}) - \mathcal{F}^{-1}(\tilde{h}_{i\beta}) \right) \\
&= g_j - \sum_{i,\beta} q_{ji\beta} \left(\mathcal{F}^{-1}(\tilde{h}_{i\beta}|_{x_1=\cdots=x_m=0}) - \mathcal{F}^{-1}(\tilde{h}_{i\beta}) \right) \\
&= g_j - \sum_{i,\beta} q_{ji\beta} \mathcal{F}^{-1}(\tilde{h}_{i\beta}|_{x_1=\cdots=x_m=0} - \tilde{h}_{i\beta})
\end{aligned}$$

と変形できる. ここで, $\tilde{h}_{i\beta}|_{x_1=\cdots=x_m=0} - \tilde{h}_{i\beta}$ の各項は $x_1, \ldots, x_m$ のいずれかで左から割り切れるので, $\mathcal{F}^{-1}(\tilde{h}_{i\beta}|_{x_1=\cdots=x_m=0} - \tilde{h}_{i\beta})$ の各項は $\partial_1, \ldots, \partial_m$ のいずれかで左から割り切れる. 従って,

$$\sum_{i,\beta} q_{ji\beta} \mathcal{F}^{-1}(\tilde{h}_{i\beta}|_{x_1=\cdots=x_m=0} - \tilde{h}_{i\beta}) = \sum_{i=1}^{m} \partial_i p_{ij}$$

と書き直せばよい. ∎

アルゴリズム 2 (非斉次部分の計算アルゴリズム)
入力 : ホロノミック左 D イデアル I の生成元.
重みベクトル $w = (w_1, \ldots, w_m, w_{m+1}, \ldots, w_n)$ で $w_1, \ldots, w_m > 0, w_{m+1} = \cdots = w_n = 0$ を満たすもの.

出力 : I の $x_1, \ldots, x_m$ についての積分イデアルの生成元 $\{g_1, \ldots, g_t\}$.
各生成元 g_j $(1 \leq j \leq t)$ に対して $g_j - \sum_{i=1}^{m} \partial_i p_{ij} \in I$ を満たす $p_{ij} \in D$.

1. アルゴリズム 1 を適用.
2. $g_j = \sum_{i,\beta} q_{ji\beta} \cdot \mathcal{F}^{-1}(h_{i\beta})$ を満たす $q_{ji\beta}$ を計算.
   $R_j = g_j - \sum_{i,\beta} q_{ji\beta} \mathcal{F}^{-1}(\tilde{h}_{i\beta})$ を $R_j = \sum_{i=1}^{m} \partial_i p_{ij}$ と書き直す.
3. p_{ij} を出力.

例 1 (積分イデアルとその非斉次部分の計算例)
積分 $A(x_2)=\int_a^b e^{-x_1^2x_2}dx_1$ の満たす非斉次微分方程式を考える. 被積分関数を $f(x_1,x_2)=e^{-x_1^2x_2}$ とする. その零化イデアルは, $I=\langle \partial_1+2x_1x_2, \partial_2+\partial_1^2\rangle$ である. 積分アルゴリズム (アルゴリズム 1) に従い, I の x_1 についての積分イデアルを計算する.
(アルゴリズム 1 の手順)

(設定) $n=2, m=1, D=K\langle x_1,x_2,\partial_1,\partial_2\rangle, D'=K\langle x_2,\partial_2\rangle, w=(1,0)$

1. I の Fourier 変換 $\mathcal{F}(I)=\langle x_1-2x_2\partial_1, \partial_2+\partial_1^2\rangle$
   (a) $\mathcal{F}(I)$ の $<_{(-w,w)}$ についてのグレブナ基底 $\mathcal{H}=\{h_1=x_1-2x_2\partial_1, h_2=\partial_2+\partial_1^2, h_3=2x_2\partial_2+x_1\partial_1+1, h_4=4x_2^2\partial_2+2x_2+x_1^2\}$
   (b) $\mathcal{F}(I)$ の $(-w,w)$ についての generic b 関数は $b(s)=s(s-1)$ であり, その非負最大整数根は $s_0=1$
   (d) $\mathcal{B}_1=\{1,\partial_1\}$, グレブナ基底の各元 h_i の $(-w,w)$ についての階数は $m_1=1, m_2=2, m_3=0, m_4=0$.
   (e) $\tilde{\mathcal{B}}=\{h_1,h_3,\partial_1h_3,h_4,\partial_1h_4\}$,
   $\mathcal{B}=\{-2x_2\partial_1, 2x_2\partial_2+1, 2x_2\partial_1\partial_2+2\partial_1, 4x_2^2\partial_2+2x_2, 4x_2^2\partial_1\partial_2+2x_2\partial_1\}$
2. $\mathcal{F}^{-1}(\mathcal{B})=\{2x_2x_1, 2x_2\partial_2+1, -2x_1x_2\partial_2-2x_1, 4x_2^2\partial_2+2x_2, -4x_1x_2^2\partial_2-2x_1x_2\}$
   $M=\{(0,2x_2),(2x_2\partial_2+1,0),(0,-2x_2\partial_2-2),(4x_2^2\partial_2+2x_2,0),(0,-4x_2^2\partial_2-2x_2)\}$
3. 左 D' 加群 $M\subset (D')^2$ の POT 項順序についてのグレブナ基底は $\{(0,x_2),(2x_2\partial_2+1,0)\}$ となるので, 積分イデアルの生成元は $g=2x_2\partial_2+1$ となる.

(アルゴリズム 2 の手順)
積分イデアルの生成元 $g=2x_2\partial_2+1$ について, 先の計算結果から, $g=\mathcal{F}^{-1}(h_3|_{x_1=0})$ である.

$$I\ni \mathcal{F}^{-1}(h_3)=g-(g-\mathcal{F}^{-1}(h_3))=g-\mathcal{F}^{-1}(h_3|_{x_1=0}-h_3)$$

だから,

$$g=(I\text{ の元})+\mathcal{F}^{-1}(h_3|_{x_1=0}-h_3)=(I\text{ の元})+\mathcal{F}^{-1}(-x_1\partial_1)=(I\text{ の元})+\partial_1x_1$$

となり, g の非斉次部分が ∂_1x_1 だとわかる.
よって, この積分 $A(x_2)$ は, $(2x_2\partial_2+1)\cdot A(x_2)=[x_1f(x_1,x_2)]_{x_1=a}^{x_1=b}$ なる非斉次微分方程式を満たすことがわかる.

多重積分についても, アルゴリズム 2 を繰り返し用いることにより, その非斉次微分方程式系を求めることが可能である.

定理 2 (多重積分の満たす微分方程式系を求めるアルゴリズム)
多重積分

$$F(x_{m+1},\ldots,x_n)=\int_{a_1}^{b_1}\cdots\int_{a_m}^{b_m} f(x_1,\ldots,x_n)dx_1\cdots dx_m \quad (m\le n)$$

を考える. 被積分関数 $f(x_1,\ldots,x_n)$ の満たす ホロノミックな微分方程式系が得られれば, 多重積分 $F(x_{m+1},\ldots,x_n)$ の満たす非斉次微分方程式が得られる.

証明 [概略] $m=1$ のときは, アルゴリズム 2 を適用する. $m \geq 2$ のときは, アルゴリズム 2 を用いて, m 個の $m-1$ 重積分に分解できるので, それを繰り返せばよい. ここでは, 簡単のため $m=2$ の場合について, 多重積分

$$F(x_3,\ldots,x_n)=\int_{a_1}^{b_1}\int_{a_2}^{b_2} f(x_1,\ldots,x_n)dx_1dx_2 \quad (2\leq n)$$

の満たす微分方程式系を求めるアルゴリズムについて説明する.

被積分関数 $f(x_1,\ldots,x_n)$ の満たすホロノミックな微分方程式系を I とし, I の x_1,x_2 についての積分イデアル

$$J=(I+\partial_1 D+\partial_2 D)\cap D' \quad (D'=K\langle x_3,\ldots,x_n,\partial_3,\ldots,\partial_n\rangle)$$

を計算する. J のある元 P をとってくると, $P=P_0+\partial_1 P_1+\partial_2 P_2$ $(P_0\in I, P_1,P_2\in D)$ と表すことができる. P を積分 F に作用させると,

$$P\cdot F=\int_{a_2}^{b_2}(P_1\cdot f|_{x_1=b_1}-P_1\cdot f|_{x_1=a_1})dx_2+\int_{a_1}^{b_1}(P_2\cdot f|_{x_2=b_2}-P_2\cdot f|_{x_2=a_2})dx_1$$

となる. ここで, 右辺の第 1 項目の積分を F_1, 被積分関数を f_1 とし, 右辺の第 2 項目の積分を F_2, 被積分関数を f_2 とする.

積分 F_1 の満たす微分方程式系を計算するために, 被積分関数 f_1 の満たすホロノミックな微分方程式系 I_1 を計算する. Oaku のアルゴリズムを用いて f_1 の零化イデアルを計算してもよいが, I から次のようにして計算することもできる.

イデアル商 $I:P_1$ は $P_1\cdot f$ を零化するホロノミックイデアルである. ($Q\in I:P_1$ とすれば $QP_1\in I$ すなわち $(QP_1)\cdot f=0$. また I がホロノミックより $I:P_1$ はホロノミック.) $P_1\cdot f|_{x_1=b_1}$ を零化する ホロノミックイデアル J_1 を得るには, $I:P_1$ の $x_1=b_1$ についての制限イデアルを計算してやればよい. 同様にして, $P_1\cdot f|_{x_1=a_1}$ を零化する ホロノミックイデアル J_2 も得られる. $f_1=P_1\cdot f|_{x_1=b_1}-P_1\cdot f|_{x_1=a_1}$ を零化するホロノミックイデアル I_1 は $J_1\cap J_2$ で得られる.

I_1 の x_2 についての積分イデアル

$$K_1=(I_1+\partial_2 D_1)\cap D' \quad (D_1=K\langle x_2,x_3,\ldots,x_n,\partial_2,\partial_3,\ldots,\partial_n\rangle)$$

を計算する. K_1 のある元 $P^{(1)}$ をとってくると, $P^{(1)}=P_0^{(1)}+\partial_2 P_2^{(1)}$ $(P_0^{(1)}\in K_1, P_2^{(1)}\in D_1)$ と表すことができ, $P^{(1)}$ を積分 F_1 に作用させると,

$$P^{(1)}\cdot F_1=P_2^{(1)}\cdot f_1|_{x_2=b_2}-P_2^{(1)}\cdot f_1|_{x_2=a_2}$$

となる. 積分 F_2 についても同様に, f_2 の零化イデアル I_2, I_2 の x_1 についての積分イデアル K_2 を計算する.

上の結果から

$$P^{(1)}\cdot P\cdot F=P^{(1)}\cdot F_1+P^{(1)}\cdot F_2$$

なる式が得られ, 右辺の第 1 項目の積分は計算できる. 第 2 項目の積分を計算するために, $K_2:P^{(1)}$ を計算し, その中の元 $P^{(2)}$ をとれば, $P^{(2)}P^{(1)}\in K_2$ となり, $P^{(2)}P^{(1)}\cdot F_2$ で計算ができる. 結局,

$$P^{(2)}P^{(1)}P\cdot F=P^{(2)}P^{(1)}\cdot F_1+P^{(2)}P^{(1)}\cdot F_2$$

により, 積分 F についての非斉次な微分方程式が得られたことになる. ∎

注意 1
非斉次微分方程式系

$$\ell_1 \cdot f = g_1, \ldots, \ell_p \cdot f = g_p \quad (\ell_i \in D', g_i \text{ はホロノミック関数.})$$

において, $\langle \ell_1, \ldots, \ell_p \rangle$ が D' におけるホロノミックイデアルとなるとき, この非斉次微分方程式系を, 非斉次ホロノミック系と呼ぶことにする. $m = 1$ のとき, 2 の出力は非斉次ホロノミック系であるが, $m \geq 2$ のとき, 出力が常に非斉次ホロノミック系であるかどうかは未解決である. しかし, 得られた微分方程式系が非斉次ホロノミック系でないときは, 後に解説する Oaku-Shiraki-Takayama アルゴリズム ([7]) を用いて, 非斉次ホロノミック系にすることができる.

4 既存のアルゴリズムとの比較

4.1 Almkvist-Zeilberger アルゴリズム

2 変数の hyperexponential function $F(t,x)$ について, 定積分 $\int_a^b F(t,x)dt$ の満たす x の非斉次微分方程式を計算するアルゴリズムとして, Almkvist-Zeilberger アルゴリズム (AZ アルゴリズム) ([1]) がある. ここで, hyperexponential function とは, 各変数について log 微分が有理関数となるような関数のことである.

アルゴリズム 3 (AZ アルゴリズム, [1])
入力: hyperexponential function $F(t,x)$

出力: x の微分作用素 $\bar{S}(x,\partial_x)$ と hyperexponential function $G(t,x)$ で次の微分方程式を満たすもの.

$$\bar{S}(x,\partial_x) \cdot F(t,x) = \partial_t \cdot G(t,x)$$

さらに $G(t,x) =$ (有理関数)$F(t,x)$ も成り立つ.

連続版の Gosper アルゴリズムと未定係数法を繰り返し用いて計算を行っている.

例 2

$$\int_0^\infty e^{-t-xt^3} dt$$

の被積分関数 $F(t,x) = e^{-t-xt^3}$ に対して AZ アルゴリズムを適用してやると,

$$\bar{S}(x,\partial_x) \cdot F(t,x) = \partial_t \cdot G(t,x)$$

が成り立つ微分作用素 $\bar{S}(x,\partial_x)$ と hyperexponential function $G(t,x)$ が得られる.

$$\bar{S}(x,\partial_x) = 27x^3\partial_x^2 + 54x^2\partial_x + 6x + 1$$

$$G(t,x) = (-9t^4x^2 + 3t^2x + 6tx - 1)F(t,x)$$

多項式 $H(t,x) = (-9t^4x^2 + 3t^2x + 6tx - 1)$ とおき, 上の式を変形すれば,

$$(\bar{S}(x,\partial_x) - \partial_t H(t,x)) \in \mathrm{Ann}_{D_2} F(t,x)$$

$\bar{S}(x,\partial_x)$ が, 零化イデアル $\mathrm{Ann}_{D_2} F(t,x)$ の t についての積分イデアル

$$(\mathrm{Ann}_{D_2} F(t,x) + \partial_t D_2) \cap D_1$$

に含まれることがわかる. (ただし, $D_2 = K\langle x, t, \partial_x, \partial_t\rangle, D_1 = K\langle x, \partial_x\rangle$ としておく.) この場合は, 積分アルゴリズムによる計算結果と全く同じ結果を返している. Risa/Asir でこのアルゴリズムを実行した結果は次の通り.

```
[1481] gosper_int_op(exp(-t-x*t^3), x,t);
[27*x^3*dx^2+54*x^2*dx+6*x+1,
-9*exp(-t^3*x-t)*t^4*x^2+(3*exp(-t^3*x-t)*t^2+6*exp(-t^3*x-t)*t)*x-exp(-t^3*x-t)]
```

4.2 Chyzak アルゴリズム

Chyzak アルゴリズム ([3], [4], [5]) は未定係数法と Ore algebra におけるグレブナ基底の理論を用いたアルゴリズムであり, Ore algebra に含まれる様々なクラスを扱うことができる. 例えば, 和や積分の満たす差分方程式や微分方程式, それらの q-アナログ等を扱える. 有理関数体係数の微分作用素環 $R := K(x,t)\langle\partial_x, \partial_t\rangle$ においては, AZ アルゴリズムの適用範囲をホロノミック関数にまで拡張したものとなっている. 以下が Chyzak アルゴリズムである.

アルゴリズム 4 (Chyzak アルゴリズム ([4], Algorithm 2.))
入力: $f(x,t)$ を零化するホロノミックイデアルの生成系 B.

出力: $P(x,\partial_x)\cdot f = \sum_i \eta_i \partial_x^i f = \partial_t[Q(x,t,\partial_x,\partial_t)\cdot f]$ を満たす作用素のペア (P,Q).

1. B のグレブナ基底 G を計算して, $R/\mathrm{Ann} f$ の基底 $\{\partial_x^\alpha \partial_t^\beta\}_{(\alpha,\beta)\in I}$ を得る.
2. $L = 0, 1, 2, \ldots$ として以下を繰り返す.
   (a) 未定係数 $\phi_{\alpha,\beta} \in K(x,t)$, $\eta_i \in K(x)$ を導入して,

$$\partial_t \sum_{(\alpha,\beta)\in I} \phi_{\alpha,\beta}\partial_x^\alpha \partial_t^\beta - \sum_{i=0}^{L} \eta_i \partial_x^i$$

   を G で割り算して, 基底 $\{\partial_x^\alpha \partial_t^\beta\}_{(\alpha,\beta)\in I}$ の一次結合に書き直す.
   (b) $\phi_{\alpha,\beta}$, η_i の連立系と見て解を計算する.
   (c) もし, 解ければ (P,Q) を返す. ただし, $P = \sum_{i=0}^{L} \eta_i \partial_x^i, Q = \sum_{(\alpha,\beta)\in I} \phi_{\alpha,\beta}\partial_x^\alpha \partial_t^\beta$.

このアルゴリズムには Chyzak 自身による実装があり, Maple のパッケージ `Mgfun` ([12]) として公開されている. このパッケージ中の関数 `creative_telescoping` は, デフォルトでは, アルゴリズム中のステップ 2 を更に繰り返して, 得られた作用素 P たちによって生成されるイデアルがホロノミックになるまで繰り返される.

計算時間を比較してみると, 一般には, グレブナ基底計算の方が未定係数法より計算コストが大きいので, 多くの問題において, Chyzak アルゴリズムの方が我々のアルゴリズムより速い. しかし, 出力の微分作用素の階数が高く項の数が少ないような問題では, 我々のアルゴリズムの方が早くなることがある. 以下にそのような例を示す.

例 3
定積分

$$F(x,y) = \int_a^b \frac{1}{xt + y + t^{10}} dt.$$

の満たす非斉次微分方程式系を計算する. 以下は, 我々のアルゴリズム (Risa/Asir のパッケージ nk_restriction) によって計算した時の出力であり, 計算にはおよそ 1.3 秒を要した.

```
[2345] load("nk_restriction.rr");
[2545] F=x*t+y+t^10$
[2546] Ann=ann(F)$ /* annihilating ideal of F^s */
0.052sec(0.0485sec)
[2547] Id=map(subst, Ann, s, -1)$ /* substitute s=-1 in Ann */
0sec(4.411e-05sec)
[1569] nk_restriction.integration_ideal(Id,[t,x,y],[dt,dx,dy],[1,0,0]|inhomo=1);
-- nd_weyl_gr :0.012sec + gc : 0.008001sec(0.02009sec)
-- weyl_minipoly :0sec(0.001189sec)
-- generic_bfct_and_gr :0.016sec + gc : 0.008001sec(0.02358sec)
generic bfct : [[1,1],[s,1],[s-9,1]]
S0 : 9
B_{S0} length : 10
-- fctr(BF) + base :0.044sec + gc : 0.024sec(0.0674sec)
-- integration_ideal_internal :0.8321sec + gc : 0.236sec(1.071sec)
[[9*x*dx+10*y*dy+9,-10*dx^9-x*dy^9,-9*dx^10+y*dy^10+9*dy^9],
[[[[dt,-t]],1],[[[dt,-dy^8]],1],[[[dt,-t*dy^9]],1]]]
0.9081sec + gc : 0.28sec(1.19sec)
```

以下は, Maple12 上で Chyzak アルゴリズム (パッケージ Mgfun [12]) によって計算したときの出力であり, 計算にはおよそ 50 秒を要した.

```
with(Mgfun):
f:=1/(x*t+y+t^10):
ts:=time():
creative_telescoping(f,[x::diff,y::diff], t::diff):
time()-ts;
                                  49.583
```

これらの計算は共に, Intel Xeon 5450 (3.00GHz), 32 GB のメモリを搭載した計算機で行った.

4.3 Oaku-Shiraki-Takayama アルゴリズム

Oaku-Shiraki-Takayama によるアルゴリズム (以降, OST アルゴリズム) は, 各有限区間 $[a,b]$ における積分を Heaviside 関数を用いて $\mathbb{R}$ 上の積分とみなして積分アルゴリズムを適用し, 積分の満たす斉次微分方程式を得るアルゴリズムである. つまり, 今 $Y(t)$ を $Y(t)=0\,(t<0), Y(t)=1\,(t\geq 0)$ で定義される関数とすると,

$$\int_a^b u(t,x)dt = \int_{-\infty}^{\infty} Y(t-a)Y(b-t)u(t,x)dt$$

であるから, 左辺の満たす斉次微分方程式を得るために, 右辺の非積分関数の満たす微分方程式系を入力として積分アルゴリズムを適用するということである. 彼らは Heaviside 関数と $u(t,x)$ の積の満たす微分方程式系を与える方法として以下の 2 種類を与えている.

(a) Heaviside 関数の性質を利用する方法

(b) D 加群のテンソル積の計算を用いる方法

前者は入力の微分方程式系を多項式倍する程度の処理で済むので計算は直ちに終了するが, その出力がホロノミックであるかどうかは分かっていない. つまり, その後の積分アルゴリズムの停止性は保証されない.

後者は $u(t,x)$ の満たすホロノミック系を入力とすれば, 積の満たすホロノミック系を得ることができるので, その後の積分アルゴリズムの停止性は保証される. しかし, テンソル積の計算のコストは非常に大きい. ここでは 前者を用いて計算する方法を OST アルゴリズム a, 後者を用いて計算する方法を OST アルゴリズム b と呼ぶことにする. 詳しくは [7, Chap 5] を参照.

例 4 ([7], Example 5.1.)

$$v(x) = \int_0^\infty e^{(-t^3+t)x} dt$$

以下は, $v(x)$ の満たす微分方程式系を OST アルゴリズムによって計算した結果である.

```
[1486] B = [dt+(3*t^2-1)*x,dx+t^3-t];
[dt+(3*t^2-1)*x,dx+t^3-t]
[1487] ost_integration_ideal(B,[t,x],[dt,dx],[1,0],[0],["inf"]);
-- nd_weyl_gr :0sec(0.014sec)
-- weyl_minipoly :0.004sec(0.000402sec)
-- generic_bfct_and_gr :0.004sec(0.0004091sec)
generic bfct : [[1,1],[s,1],[s-2,1]]
S0 : 2
B_{S0} length : 3
-- fctr(BF) + base :0sec(0.000891sec)
-- integration_ideal_internal :0sec(0.000793sec)
[-27*x^3*dx^3-54*x^2*dx^2+(4*x^3+3*x)*dx+4*x^2-3,
27*x^2*dx^4+135*x*dx^3+(-4*x^2+105)*dx^2-16*x*dx-8]
```

また, 以下は我々のアルゴリズムによって計算した結果である.

```
[1567] B = [dt+(3*t^2-1)*x,dx+t^3-t];
[dt+(3*t^2-1)*x,dx+t^3-t]
[1568] INT=integration_ideal(B,[t,x],[dt,dx],[1,0]|inhomo=1);
-- nd_weyl_gr :0.004001sec(0.004164sec)
-- weyl_minipoly :0sec(0.002094sec)
-- generic_bfct_and_gr :0.008002sec(0.007762sec)
generic bfct : [[1,1],[s,1],[s-2,1]]
S0 : 2
B_{S0} length : 3
-- fctr(BF) + base :0.004sec(0.003639sec)
-- integration_ideal_internal :0.008sec(0.005342sec)
[[-27*x^2*dx^2-27*x*dx+4*x^2+3],[[[[dt,-9*t*x*dx+(-6*t^2+4)*x+3*t]],1]]]
```

我々のアルゴリズムの出力からは $v(x)$ の満たす非斉次微分方程式系として

$$(-27x^2\partial_x^2 - 27x\partial_x + 4x^2 + 3) \cdot v(x) = -4x \tag{3}$$

が得られる. 一方, $-4x$ の零化イデアルは $\langle x\partial_x - 1, \partial_x^2 \rangle$ であるから, この 2 つの生成元を (3) の両辺に左から作用させると $v(x)$ の満たす斉次微分方程式系が得られ, OST アルゴリズムによる出力と一致する. しかし, OST アルゴリズムの出力からは右辺の非斉次部分を直接計算することは出来ない. つまり, 我々のアルゴリズムは OST アルゴリズムよりも多くの情報を含んでいる.

表 1 は

$$v_n(x) = \int_0^\infty u_k(t,x)dt,\ \bar{v}_n(x) = \int_0^1 u_k(t,x)dt \quad (k = 1, \ldots, 4)$$

$$\text{ただし,} u_k(t,x) = \exp\left(-tx\prod_{i=1}^{k}(t^2-i^2)\right)$$

に対して, 我々のアルゴリズム (アルゴリズム 2) と OST アルゴリズム a, b を適用したときの計算時間を表している. また, 比較の為に アルゴリズム 1 の計算時間も示した. 計算環境は CPU:Xeon X5570 (2.93GHz), Memory:48GB である. 記号 † は v_i の結果を利用できるので, 改めて計算する必要がないことを示している.

	Alg. 2			OST a	OST b			Alg. 1
input	int	Ann	total	total	prod	int	total	total
v_1	0.0042	0.0014	0.0056	0.0062	0.11	0.012	0.12	0.0039
v_2	0.15	0.019	0.17	0.25	5.10	0.16	5.26	0.075
v_3	19.91	0.45	20.36	96.14	24.54	95.24	119.8	13.58
v_4	26724	28.33	26752	> 1 day	1726	> 1 day	—	24003
$\bar{v}_1$	†	0.0015	0.0057	0.0071	0.47	0.0050	0.48	n/a
$\bar{v}_2$	†	0.027	0.18	1.56	18230	1.19	18231	n/a
$\bar{v}_3$	†	1.62	21.53	3769	848	2802	3650	n/a
$\bar{v}_4$	†	294	27018	> 1 day	16231	> 1 day	—	n/a

表 1: OST アルゴリズムとの計算時間 (sec) の比較

計算効率の面では, アルゴリズム 2 はアルゴリズム 1 と比べて, 元々計算しているグレブナ基底の中間情報を用いて非斉次部分を構成するコストが増える. 一方, OST アルゴリズムは, 非積分関数に Heaviside 関数をかけることで積分アルゴリズムの入力イデアルが複雑になり計算コストが増加する. コストの増加分は, 単に微分作用素の積や和をとるだけの計算であるアルゴリズム 2 の方が小さいが, OST アルゴリズムの出力のように斉次微分方程式を導出する為には, 非斉次部分の零化イデアルの計算が必要である. 零化イデアルの計算は重たい場合もあり, そのコストも含めると両者のどちらが計算に有利であるかは問題に依存する.

同一の非積分関数について積分区間を変えたものの満たす微分方程式を得たい場合, OST アルゴリズムでは初めから計算をやり直す必要があるが, 我々のアルゴリズムは積分区間に依存しないのでその必要はない.

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] M. Apagodu, D. Zeilberger, Multi-variable Zeilberger and Almkvist-Zeilberger algorithms and the sharpening of Wilf-Zeilberger theory, Advances in Applied Mathematics **37**, 139-152, 2006.

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

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

[5] F. Chyzak, B. Salvy, Non-commutative Elimination in Ore Algebras Proves Multivariate Holonomic Identities, Journal of Symbolic Computation **26**, 187-227, 1998.

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

[7] T. Oaku, Y. Shiraki, 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.

[8] M. Saito, B. Sturmfels, N. Takayama, *Gröbner Deformations of Hypergeometric Differential Equations*, Springer, 2000.

[9] N. Takayama, An Approach to the Zero Recognition Problem by Buchberger Algorithm, Journal of Symbolic Computation **14**, 265–282, 1992.

[10] A. Tefera, MultInt, a Maple package for multiple integration by the WZ method, Journal of Symbolic Computation **34**, 329-353, 2002.

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

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

[13] C. Koutschan: `HolonomicFunctions`,
`http://www.risc.jku.at/research/combinat/software/HolonomicFunctions/`

[14] H. Nakayama, K. Nishiyama: Risa/Asir パッケージ `nk_restriction.rr`,
`http://www.math.kobe-u.ac.jp/~nakayama/nk_restriction.rr`