数理解析研究所講究録 1456 巻 2005 年 133-143

高速留数計算アルゴリズム

庄司卓夢
TAKUMU SHOJI
新潟大学自然科学研究科
GRADUATE SCHOOL OF SCIENCE AND TECHNOLOGY, NIIGATA UNIVERSITY
田島慎一
SHINICHI TAJIMA
新潟大学工学部情報工学科
FACULTY OF ENGINEERING, NIIGATA UNIVERSITY

Abstract

留数は古典的な対象であるが, 計算機代数の分野ではその計算法についてそれほど注目されていなかった. 本研究では, 代数解析の観点から, 有理関数が内部に隠しもつ数学的構造を明らかにし, D 加群の理論を用いて効率的な留数計算アルゴリズムを導出する. そして, そのアルゴリズムを数式処理システムに実装し, 性能を評価する.

1 はじめに

留数 $\mathrm{Res}_\alpha(\frac{h}{f^\ell}) = \frac{1}{2\pi i}\oint \frac{h}{f^\ell}dx$ を求める問題を考える. ただし, $f, h \in K[x]$, $K = \mathbb{Q}$, f は既約, $\ell \in \mathbb{N}$, $\alpha \in Z = \{x \in \mathbb{C} | f(x) = 0\}$ とする.

Fact ([2, 4]) 有理関数 $\frac{1}{f^\ell}$ が定める代数的局所コホモロジー類 $[\frac{1}{f^\ell}] \in H^1_{[Z]}(K[x])$ は, $\ell - 1$ 階の微分作用素 T を用いて次のように表せる.

$$[\frac{1}{f^\ell}] = T[\frac{f'}{f}],\ T = \sum_{i=0}^{\ell-1}(-\frac{d}{dx})^{\ell-1-i}t_i,\ t_i \in K[x]/\langle f\rangle.$$

従って, T の形式随伴を T^* とおくと,

$$\oint \frac{h}{f^\ell}dx = \oint h[\frac{1}{f^\ell}]dx = \oint hT[\frac{f'}{f}]dx = \oint hT(\frac{f'}{f})dx = \oint T^*h(\frac{f'}{f})dx$$

となり, 次式が導かれる.

Theorem 1.1

$$\mathrm{Res}_\alpha(\frac{h}{f^\ell}) = (T^*h)(\alpha).$$

留数はこの式で exact に求まる.

本稿では, まず次節でこの T の計算方法について解説する. 第 3 節では, 一般的な有理関数の場合にまで理論を拡張する. 第 4 節では, 留数計算アルゴリズムを具体的に与える. 第 5 節では, 具体例を紹介する. 第 6 節では, 計算機実験によって各アルゴリズムの計算効率を比較し, 性能を評価する.

2 微分作用素Tの計算方法

Tを求める方法として, 次の3通りの方法が考えられる.

	アルゴリズムの概略	プログラム名
方法1	連立方程式を解く (線形代数)	laurent1
方法2	微分作用素の合成 + 拡張 Euclid 互除法	laurent2
方法3	微分方程式の理論を用いる	snoether

方法1と方法2については文献 [4] に概略が述べてある. 本稿は, 方法3を解説していく.

Fact ([1, 3, 4])

$D_X = K[x, \frac{d}{dx}]$, $\sigma = [\frac{1}{f^\ell}]$ とおき, 微分作用素 P を $P = f\frac{d}{dx} + \ell f'$ で定めると次が成り立つ.

・$\mathrm{Ann}_{D_X}(\sigma) = D_X\langle P, f^\ell\rangle$.

左 D_X 加群 M, N を, $M = D_X/\mathrm{Ann}_{D_X}(\sigma)$, $N = D_X/D_X\langle f\rangle$ で定めると次が成り立つ.

・$\dim_K \mathrm{Hom}_{D_X}(M, N) = \deg\ f$.

・$\mathrm{Hom}_{D_X}(M, N)$ は, 右 $K[x]/\langle f\rangle$ 加群の構造をもつ.

文献 [1] pp.52-53 によると, T の各右係数 t_i は, t_0 と $(f')^{-1}$ のべきを共通因子としてもつ. それに注目すれば, 計算を効率化できる.

$B_i = (-\frac{d}{dx})^i (f')^i$ とおくと, $\mathrm{Hom}_{D_X}(D_X/D_X\langle f^\ell\rangle, N)$ は $B_0, B_1, \ldots, B_{\ell-1}$ で生成される. そこで,

$$B = \sum_{i=0}^{\ell-1} B_{\ell-1-i} c_i,\ c_i \in K[x]/\langle f\rangle,\ c_0 = 1$$

なる $B \in \mathrm{Hom}_{D_X}(M, N)$ を考える.

Lemma 2.1 各 c_i は次の漸化式で求まる. $i = 1, 2, \ldots, \ell-1$ に対して,

$$c_0 = 1,\ c_i = -\frac{1}{i}\sum_{j=1}^{i}\{\prod_{k=1}^{j}(\ell - 1 - i + k)\}\frac{(\ell j + i - j)}{(j+1)!} f^{(j+1)}(f')^{j-1} c_{i-j}.$$

証明: $PB = P\sum_{i=0}^{\ell-1} B_{\ell-1-i}c_i = \sum_{i=0}^{\ell-1} B_{\ell-1-i} w_{\ell-1-i}$ とおく. ここで,

$$\begin{aligned} PB_{\ell-1-i} &= (-f)(-\frac{d}{dx})B_{\ell-1-i} + \ell f' B_{\ell-1-i} \\ &= \sum_{k=0}^{\ell-1-i} B_{\ell-1-i-k}\{(\ell k - i)\frac{(\ell-1-i)!}{(\ell-1-i-k)!(k+1)!}\} f^{(k+1)}(f')^k \end{aligned}$$

を用いて $w_{\ell-1-i}$ を求めると,

$$w_{\ell-1-i} = \sum_{j=0}^{i}(\ell j + i - j)\frac{(\ell-1-i+j)!}{(\ell-1-i)!(j+1)!} f^{(j+1)}(f')^j c_{i-j}$$

となる. $PB \in D_X f$ より 各 i に対し, $w_{\ell-1-i} \equiv 0 \pmod f$ が成り立つので,

$$i c_i + \sum_{j=1}^{i}(\ell j + i - j)\frac{(\ell-1-i+j)!}{(\ell-1-i)!(j+1)!} f^{(j+1)}(f')^{j-1} c_{i-j} = 0$$

が得られる. ∎

Lemma 2.2 $T = Bu$ となる $u \in K[x]/\langle f \rangle$ は, 条件 $(\ell-1)!(f')^{2\ell-1}u = 1$ で与えられる.

証明: $T = Bu$ とおく. $PT = PBu \equiv 0 \pmod{D_X f}$ が成り立つので, 代数的局所コホモロジー類 $\tau = T[\frac{f'}{f}]$ は微分方程式系 $P\tau = 0$, $f^\ell \tau = 0$ を満たす.
ここで, $[\frac{1}{f^\ell}] = T[\frac{f'}{f}]$ の両辺に $f'f^{\ell-1}$ をかけ, $f^{\ell-1}B \equiv (\ell-1)!(f')^{2\ell-2} \pmod{D_X f}$ を用いて計算していくと,

$$(\ell-1)!(f')^{2\ell-1}u \equiv 1 \pmod{f}$$

が得られる. 従って, $f'(f^{\ell-1})\tau = [\frac{f'}{f}]$ が成り立つ. $P\tau = 0$, $f^\ell\tau = 0$ の解で, $f'(f^{\ell-1})\tau = [\frac{f'}{f}]$ を満たすものは σ だけだから, $\tau = \sigma$ が証明された. ∎

Lemma 2.3 $T^*h = u(B^*h)$ が成り立つ.

証明: $T^* = (Bu)^* = u^*B^* = uB^*$ ∎

従って, Theorem 1.1 は次のように変形できる.

Theorem 2.4

$$\mathrm{Res}_\alpha(\frac{h}{f^\ell}) = (u(B^*h))(\alpha).$$

3 一般化

一般的な有理関数 $\frac{h}{f_1^{\ell_1}\cdots f_m^{\ell_m}}$ の場合を考える. ただし, $f_1, \ldots, f_m \in K[x]$ は既約とする.
まず, $Z = \{x \in \mathbb{C} | f_1(x)\cdots f_m(x) = 0\}$ に台を持つ代数的局所コホモロジー類 $\sigma = [\frac{1}{f_1^{\ell_1}\cdots f_m^{\ell_m}}]$ を考える. $p = -f_1f_2\cdots f_m$, $q = \ell_1 f_1' f_2\cdots f_m + \ell_2 f_1 f_2' \cdots f_m + \cdots + \ell_m f_1 f_2 \cdots f_m'$ を用いて, 微分作用素 P を $P = p(-\frac{d}{dx}) + q$ で定めると, σ の満たす微分方程式系 $P\sigma = 0$, $(f_1^{\ell_1}\cdots f_m^{\ell_m})\sigma = 0$ を得る.
次に, σ の直和分解を $\sigma = \sigma_{f_1} + \cdots + \sigma_{f_m}$, $\sigma_{f_k} \in H^1_{[Z_{f_k}]}(K[x])$, $Z_{f_k} = \{x | f_k(x) = 0\}$ とおく. 一般に, 微分作用素は local operator であるので, $P\sigma = 0$ より $P\sigma_{f_k} = 0$ が従う. また, D_X 加群 $M = D_X/D_XP + D_X(f_1^{\ell_1}\cdots f_m^{\ell_m})$ は各点で simple なので, $\sigma_{f_k} = T_{f_k}[\frac{f_k'}{f_k}]$ なる T_{f_k} を求めることができる.
(有理関数 $\frac{h}{f_1^{\ell_1}\cdots f_m^{\ell_m}}$ の部分分数分解を求めずに直接構成できる.)

今, $f = f_k$, $\ell = \ell_k$, $a = f_1\cdots f_{k-1}f_{k+1}\cdots f_m$ とおく.

前節と同様に $B_i = (-\frac{d}{dx})^i(f'a)^i$ とおき, $B_f = \sum_{i=0}^{\ell-1} B_{\ell-1-i}c_i$ を考える.

Lemma 3.1 各 c_i は次の漸化式で求まる. $i = 1, 2, \ldots, \ell-1$ に対して,

$$c_0 = 1,\ c_i = -\frac{1}{i}\sum_{j=1}^{i}\{\binom{\ell-i+j}{j+1}p^{(j+1)} + \binom{\ell-1-i+j}{j}q^{(j)}\}(f'a)^{j-1}c_{i-j}.$$

証明: Lemma 2.1 と同様. ∎

Lemma 3.2 $g = f_1^{\ell_1}\cdots f_{k-1}^{\ell_{k-1}} f_{k+1}^{\ell_{k+1}}\cdots f_m^{\ell_m}$ とおく. $T_f = B_f u$ となる $u \in K[x]/\langle f \rangle$ は,
条件 $(\ell-1)!(f')^{2\ell-1}a^{\ell-1}gu = 1$ で与えられる.

証明: Lemma 2.2 と同様. ∎

4 留数計算アルゴリズム

2節での論述から, 有理関数 $\frac{h}{f^\ell}$ に対して, 次の留数計算アルゴリズムを得る.

Algorithm 4.1 compute Res($\frac{h}{f^\ell}$)

input f, ℓ, h
step1 $c_0, c_1, \ldots, c_{\ell-1}$ (lemma 2.1)
step2 $B^*h = \sum_{i=0}^{\min\{\ell-1, \deg h\}} (f')^i c_{\ell-1-i} h^{(i)}$
step3 $Q \leftarrow (f')^{-1}$
step4 $u = \frac{1}{(\ell-1)!} Q^{2\ell-1}$
step5 Res $= u(B^*h)$
output Res

アルゴリズムの要点を簡単に述べる. 骨格は, Theorem 2.4 に基づいて設計され, T の計算は Lemma 2.1, 2.2, 2.3 により効率化している.

また, 各 step の計算は全て剰余体 $K[x]/\langle f \rangle$ で行えるため, 多項式の次数の増大を防ぎ, 逆元計算が利用できる. この性質は, 高速化に強烈に効いてくる.

プログラムでは, できる限り $\mathbb{Z}[x]$ での計算に問題を帰着させると計算効率が上がる.

3節での論述から, 一般的な有理関数に対して, 次の留数計算アルゴリズムを得る.

Algorithm 4.2 compute Res($\frac{Num}{Den}$)

input Num, Den
step0 簡約化 ($\frac{Num}{Den} \rightarrow \frac{h}{f_1^{\ell_1} \cdots f_m^{\ell_m}}$)
for k $\leftarrow$ 1 **to** m {
  $f \leftarrow f_k$, $\ell \leftarrow \ell_k$, $a \leftarrow \prod_{i=1, i\neq k}^{m} f_i$, $g \leftarrow \prod_{i=1, i\neq k}^{m} f_i^{\ell_i}$
  step1 $c_0, c_1, \ldots, c_{\ell-1}$ (lemma 3.1)
  step2 $B_f{}^*h = \sum_{i=0}^{\min\{\ell-1, \deg h\}} (f'a)^i c_{\ell-1-i} h^{(i)}$
  step3 $Q_1 \leftarrow (f')^{-1}$, $Q_2 \leftarrow a^{-1}$, $Q_3 \leftarrow g^{-1}$
  step4 $u = \frac{1}{(\ell-1)!} Q_1^{2\ell-1} Q_2^{\ell-1} Q_3$
  step5 Res $= u(B_f{}^*h)$
  output Res
}

分母の各因子に対する留数計算は, 剰余体 $K[x]/\langle f_k \rangle$ で行うことができる. プログラムでは, 入力には因数分解されたデータをリストで渡すと step0 が楽になる.

各アルゴリズムに対する留数計算プログラムは次の通りである. 6節でその性能を見ている.

	説明	プログラム名
Alg 4.1	有理関数 $\frac{h}{f^\ell}$ の留数計算	residue
Alg 4.2	一般的な有理関数の留数計算	residue_g

5 具体例

Example 5.1 $f=x^3-x-1$, $\ell=3$, $h=1$ の時, 次の微分作用素 L を求めよ.

$$[\frac{h}{f^\ell}]=L[\frac{f'}{f}],\ L=\sum_{i=0}^{\ell-1}(-\frac{d}{dx})^{\ell-1-i}a_i$$

```
[156] laurent1(x^3-x-1,3,1);
kenzan : ok
[9/529*x^2-27/1058*x+11/1058,-81/529*x^2-9/529*x+135/529,-4905/12167*x^2+4563/12
167*x+3270/12167]
[157] laurent2(x^3-x-1,3,1);
[24334,[414*x^2-621*x+253,-3726*x^2-414*x+6210,-9810*x^2+9126*x+6540]]
[158] snoether(x^3-x-1,3,1);
[24334,[414*x^2-621*x+253,-3726*x^2-414*x+6210,-9810*x^2+9126*x+6540]]
```

答えは, $L=\frac{1}{24334}\{(-\frac{d}{dx})^2(414x^2-621x+253)+(-\frac{d}{dx})(-3726x^2-414x+6210)+(-9810x^2+9126x+6540)\}$ である.

一般に, $Z=\{x\in\mathbb{C}|f(x)=0\}$ とおくと,
$(-\frac{d}{dx})^k a(x)[\frac{f'}{f}]=(-\frac{d}{dx})^k a(x)\sum_{\alpha\in Z}[\frac{1}{x-\alpha}]=(-\frac{d}{dx})^k\sum_{\alpha\in Z}[\frac{a(\alpha)}{x-\alpha}]=\sum_{\alpha\in Z}[\frac{k!a(\alpha)}{(x-\alpha)^{k+1}}]$
となるから, 有理関数 $\frac{h}{f^\ell}$ の極における Laurent 展開の主要部は, 上記の微分作用素 L から直ちに計算できる.

この例の場合, 有理関数 $\frac{1}{(x^3-x-1)^3}$ の極 α における主要部は, 求めた L から,
$\frac{18\alpha^2-27\alpha+11}{23^2(x-\alpha)^3}+\frac{-81\alpha^2-9\alpha+135}{23^2(x-\alpha)^2}+\frac{-4905\alpha^2+4563\alpha+3270}{23^3(x-\alpha)}$ と分かる.

Example 5.2 関数 $R(x)=\frac{1}{(x^4-3x^3+2x^2+x-7)^3}$ の留数を求めよ.

微分作用素 $B=B_2c_0+B_1c_1+B_0c_2$ を考え, Alg 4.1 に従って留数を求めていく.

Input $f=x^4-3x^3+2x^2+x-7$, $\ell=3$, $h=1$

Step1 c_0,c_1,c_2 の計算 (Lemma 2.1)
$c_0=1$, $c_1=-36x^2+54x-12$, $c_2=330x^2-720x+2418$

Step2 $B^*h=\sum_{i=0}^{\min\{\ell-1,\deg h\}}(f')^i c_{\ell-1-i}h^{(i)}$ の計算
$B^*h=(f')^0c_2h^{(0)}=330x^2-720x+2418$

Step3 $Q=(f')^{-1}$ の計算
$(f')^{-1}=-\frac{137}{33090}x^3+\frac{397}{33090}x^2+\frac{73}{2206}x-\frac{556}{16545}$

Step4 $u=\frac{1}{(\ell-1)!}Q^{2\ell-1}$ の計算
分数計算はできるだけ避けたいので, 整数係数化処理を施す.
$u=\frac{1}{289854661032000}(-27289583x^3+86271673x^2+50233785x-132913868)$

Step5 $\mathrm{Res}=u(B^*h)$ の計算
$\mathrm{Res}=\frac{1}{289854661032000}(-30209148384x^3+95377636704x^2+133022875680x-173675480064)$

```
[109] residue(x^4-3*x^3+2*x^2+x-7,3,1);
[289854661032000,-30209148384*x^3+95377636704*x^2+133022875680*x-173675480064]
```

$\alpha \in \{x|f(x)=0\}$ とおくと, 留数は次のようになる.
$\mathrm{Res}_\alpha(R) = \frac{1}{18\cdot 5^3\cdot 1103^3}(-314678629\alpha^3+993517049\alpha^2+1385654955\alpha-1809119584)$

Example 5.3 関数 $R(x)=\frac{x^3+1}{x^{18}-2x^{14}+x^{10}-x^8+2x^4-1}$ の留数を求めよ.

簡約化すると, $R=\frac{x^2-x+1}{(x^4+x^3+x^2+x+1)(x^4-x^3+x^2-x+1)(x^2+1)^2(x+1)^2(x-1)^3}$ となる. Alg 4.2 によって, 部分分数分解を行わずに留数を求めることができる.

```
[118] residue_g(x^3+1,x^18-2*x^14+x^10-x^8+2*x^4-1);
x^4+x^3+x^2+x+1 : [50,-2*x^2-x-2]
x^4-x^3+x^2-x+1 : [50,-2*x^3+4*x^2-x-2]
x^2+1 : [128,32*x+20]
x+1 : [3200,-390]
x-1 : [64000,13400]
```

$\alpha \in \{x|x^4+x^3+x^2+x+1=0\}$, $\beta \in \{x|x^4-x^3+x^2-x+1=0\}$, $\gamma \in \{x|x^2+1=0\}$ とおくと, 留数は次のようになる.
$\mathrm{Res}_\alpha(R) = -\frac{1}{25}\alpha^2-\frac{1}{50}\alpha-\frac{1}{25}$, $\mathrm{Res}_\beta(R) = -\frac{1}{25}\beta^3+\frac{2}{25}\beta^2-\frac{1}{50}\beta-\frac{1}{25}$, $\mathrm{Res}_\gamma(R) = \frac{1}{4}\gamma+\frac{5}{32}$,
$\mathrm{Res}_{-1}(R) = -\frac{39}{320}$, $\mathrm{Res}_1(R) = \frac{67}{320}$

6 計算機実験

CPU 時間 (秒) を計測して, アルゴリズムの性能評価, 計算効率の比較を行う.

実験環境

CPU:Pentium4 2.6GHz, RAM:1GB, OS:WindowsXP, Software:Risa/Asir

実験で使う既約多項式

$f_1=x+1$, $f_2=x^2-3x+5$, $f_3=x^3-x-1$, $f_4=x^4+1$, $f_5=x^5+3x^4+1$, $f_6=x^6+5x^4+x^2+3$,
$f_8=x^8+5x^5-x^4-x^2+1$, $f_{10}=x^{10}+x^7-x^5+3x^4+2x+1$, $f_{25}=x^{25}+x^{12}+1$, $f_{50}=x^{50}-4x^{11}+1$

6.1 微分作用素の計算効率の比較

次のような微分作用素 L を, 2 節で述べた 3 通りの方法で計算し, 計算効率を比較する.

$$[\frac{h}{f^\ell}] = L[\frac{f'}{f}],\ L=\sum_{i=0}^{\ell-1}(-\frac{d}{dx})^{\ell-1-i}a_i$$

• 入力 $f = f_5, h = 1$

ℓ	分母の総次数	laurent1	laurent2	snoether (2004)	snoether (2005)
1	5	0	0	0	0
5	25	0.047	0	0	0
10	50	0.671	0	0	0
15	75	6.265	0.016	0.016	0
20	100	40.84	0.047	0.016	0
100	500		1.813	0.563	0.047
500	2500			46.72	3.109

snoether(2004) は, 文献 [1] pp.53-54 で述べてあるアルゴリズムである. それの Noether 作用素の構成方法を改良したものが snoether(2005) である.

6.2 有理関数 $\frac{h}{f^\ell}$ の留数計算アルゴリズムの性能評価

留数計算プログラム residue の性能評価. 各 step ごとに計算時間を見る. 留数計算用に特化させた residue は, snoether(2005) よりも高速であることが分かる.

入力	$\frac{1}{(f_5)^{500}}$	$\frac{1}{(x^{10}+2)^{250}}$	$\frac{x^{512}+x^{256}+x^{128}+x^{64}+x^{32}+1}{(f_{10})^{250}}$	$\frac{1}{(f_{25})^{100}}$	$\frac{x^{1000}}{(f_{50})^{50}}$
step1	0.219	0.016	0.703	1.906	0.688
step2	0	0	1.250	0	0.078
step3	0	0	0	0	0
step4	0.031	0	0.078	0.234	1.313
step5	0	0	0.016	0.078	0.250
計	0.250	0.016	2.047	2.219	2.328

6.3 有理関数 $\frac{1}{Den}$ の部分分数分解の計算効率の比較

・pfd1 : 未定係数法 (線形代数)
・pfd2 : 拡張 Euclid 互除法
・pfd3 : 微分方程式の理論を用いる (文献 [1] pp.55-56)

Den	総次数	pfd1	pfd2	pfd3
$f_2 f_3$	5	0	0	0
$(f_2)^3(f_3)^3(f_5)^2$	25	0.031	0	0
$(f_2)^2(f_6)^3 f_8 (f_{10})^2$	50	0.234	0.016	0.016
$f_2 f_3 (f_5)^2 (f_6)^4 (f_8)^2 (f_{10})^2$	75	1.234	0.047	0.063
$(f_2)^3(f_3)^3(f_5)^3(f_6)^4(f_8)^2(f_{10})^3$	100	4.234	0.094	0.109
$(f_2)^{12}(f_3)^{10}(f_5)^6(f_6)^{10}(f_8)^2(f_{10})^4$	200	285.4	4.062	1.313
$(f_2)^{18}(f_3)^{15}(f_5)^{17}(f_6)^{10}(f_8)^3(f_{10})^5$	300		34.09	3.766

6.4 有理関数 $\frac{1}{Den}$ の留数計算アルゴリズムの性能評価

簡約計算が重いので, プログラムにはあらかじめ因数分解されたデータをリストで渡している.

・pfd3_res : 部分分数分解を pfd3 で求め residue に渡す
・residue_g : Alg 4.2 (部分分数分解を行わない)

Den	総次数	pfd3_res	residue_g
$(f_2)^2(f_6)^3f_8(f_{10})^2$	50	0.063	0
$(f_2)^3(f_3)^3(f_5)^3(f_6)^4(f_8)^2(f_{10})^3$	100	0.422	0
$(f_2)^{12}(f_3)^{10}(f_5)^6(f_6)^{10}(f_8)^2(f_{10})^4$	200	6.141	0.016
$(f_2)^{18}(f_3)^{15}(f_5)^{17}(f_6)^{10}(f_8)^3(f_{10})^5$	300	26.03	0.047
$(f_2)^{50}(f_3)^{50}(f_4)^{40}(f_5)^{30}(f_6)^{30}(f_8)^{20}(f_{10})^{10}$	1000		0.438
$(f_1)^{1000}(f_2)^{250}(f_3)^{200}(f_4)^{200}(f_5)^{80}(f_6)^{75}(f_8)^{75}(f_{10})^{65}$	5000		16.77

7 まとめ

- 留数計算の式を, 微分作用素を使って表現した.
- 有理関数の極における Laurent 展開の主要部の計算は, 微分方程式の理論で導出した漸化式で計算すると効率的に計算できる.
- 本稿で提案した留数計算アルゴリズムの特徴.
  - exact な計算である.
  - 剰余体 $K[x]/\langle f\rangle$ における四則演算のみで計算できる.
  - 部分分数分解を行わずに直接計算できる.
  - 高速である. (特に, 高い位数を持つ極における計算が速い.)

参 考 文 献

[1] 加藤涼香, 田島慎一 : 有理関数のローラン展開アルゴリズムと代数的局所コホモロジー, 京都大学数理解析研究所講究録 1395 「Computer Algebra-Design of Algorithms, Implementations and Applications」 (2004), 50–56.

[2] 田島慎一 : 代数的局所コホモロジー類のローラン展開と L.Ehrenpreis の Noether 作用素, 京都大学数理解析研究所講究録 1138 「数式処理における理論と応用の研究」 (2000), 87–95.

[3] 田島慎一 : 確定特異点型ホロノミック系の零次元代数的局所コホモロジー解, 京都大学数理解析研究所講究録 1336 「双曲形方程式と非正則度」 (2003), 121-132.

[4] 田島慎一 : Holonomic な定数係数線形偏微分方程式系と Grothendieck duality, 京都大学数理解析研究所講究録 「積分核の代数解析的研究」 掲載予定.

[5] 田島慎一 : 一変数留数計算アルゴリズムについて, 京都大学数理解析研究所講究録 「積分核の代数解析的研究」 掲載予定.

付録 プログラム

数式処理システムは Risa/Asir (http://www.math.sci.kobe-u.ac.jp/Asir/) を使用しました.

1. snoether

```
/* F:分母 (既約多項式), L:極の位数 (自然数), H:分子 (多項式) */

def snoether(F,L,H){

  X=var(F);
  M=(L<deg(F,X)+1)?L:deg(F,X)+1;

  DF=newvect(M+1,[F]);for(I=1;I<M+1;I++) DF[I]=diff(DF[I-1],X);
  DFP=newvect(L,[1]);for(I=1;I<L;I++) DFP[I]=srem(DFP[I-1]*DF[1],F);
  K=1;C0=newvect(M,[1]);for(I=1;I<M;I++){K*=I+1;C0[I]=srem(DF[I+1]*DFP[I-1]/K,F);}

  C=newvect(L,[1]);
  for(I=1;I<L;I++){
    K=L-1-I;C1=1;
    for(J=1;J<I+1&&J<M;J++){
      C1*=K+J;
      C[I]+=C1*(L*J+I-J)*C0[J]*C[I-J];
    }
    C[I]=-srem(C[I],F)/I;
  }
  for(I=0;I<L;I++) C[I]=srem(C[I]*DFP[L-1-I],F);

  if(type(H)==1||type(H)==0){
    C*=H;
  }else{
    N=(L<deg(H,X)+1)?L:deg(H,X)+1;
    DH=newvect(N,[H]);for(I=1;I<N;I++) DH[I]=diff(DH[I-1],X)/I;DH=map(srem,DH,F);
    LC=newvect(L,[C[0]*DH[0]]);
    for(I=1;I<L;I++){
      U=C[I]*DH[0];K=L-1-I;C1=1;
      for(J=1;J<I+1&&J<N;J++){
        C1*=K+J;
        U+=C1*C[I-J]*DH[J];
      }
      LC[I]=srem(U,F);
    }
    C=LC;
  }

  Q=inversePolyn(DF[1],F);QN=ptozp(Q);
  A=QN;LP=2*L-1;K=1;UN=1;ZD=1;
  while(1){
    if(LP%2==1){Z=srem(UN*A,F);UN=ptozp(Z);ZD*=sdiv(UN,Z)*K;}
    LP=idiv(LP,2);if(LP==0) break;
    B=srem(A*A,F);A=ptozp(B);K=sdiv(A,B)*K*K;
  }
  UD=fac(L-1)*sdiv(QN,Q)^(2*L-1)*ZD;

  C=map(srem,C*UN,F);

  return [UD,vtol(C)];
}
end$
```

2. residue_g

```
/* NUM:多項式, DEN:多項式 または 因数分解されたリスト. [[因子, 重複度],...] の形. */
/* (リストで入力する場合の注意) 既約チェック, 有理式の約分, 整数係数化は行わない. 呼び出し側が責任を持つ. */

def residue_g(NUM,DEN){

  /* 簡約化 */
  if(type(DEN)==4){
    Num=NUM;
    D=cons([1,1],DEN);
  }else{
    Gcd=gcd(NUM,DEN);
    Num=sdiv(NUM,Gcd);
    Den=sdiv(DEN,Gcd);
    D=fctr(Den);
  }

  X=var(D[1][0]);
  Len=length(D);

  P=-1;Q=0;
  for(I=1;I<Len;I++){
    P*=D[I][0];
    Tmp=D[I][1];
    for(J=1;J<Len;J++) Tmp*=(I==J)?diff(D[J][0],X):D[J][0];
    Q+=Tmp;
  }

  MaxL=0;for(I=1;I<Len;I++) if(MaxL<D[I][1]) MaxL=D[I][1];
  DP=newvect(MaxL+1,[P]);for(I=1;I<MaxL+1;I++) DP[I]=diff(DP[I-1],X)/I;
  DQ=newvect(MaxL,[Q]);for(I=1;I<MaxL;I++) DQ[I]=diff(DQ[I-1],X)/I;

  H=Num;DegH=deg(H,X)+1;N=(MaxL<DegH)?MaxL:DegH;
  DH=newvect(N,[H]);for(I=1;I<N;I++) DH[I]=diff(DH[I-1],X);

  for(II=1;II<Len;II++){

    F=D[II][0];
    L=D[II][1];
    DF=diff(F,X);

    R=srem(DF*sdiv(-P,F),F);
    RP=newvect(L,[1]);for(I=1;I<L;I++) RP[I]=srem(RP[I-1]*R,F);

    DP2=newvect(L+1);for(I=2;I<L+1;I++) DP2[I]=srem(DP[I],F);
    DQ2=newvect(L);for(I=1;I<L;I++) DQ2[I]=srem(DQ[I],F);

    C=newvect(L,[1]);
    for(I=1;I<L;I++){
      K=L-1-I;KK=L-I;C1=1;
      for(J=1;J<I+1;J++){
        C1*=K+J;
        C[I]+=((KK+J)*DP2[J+1]+DQ2[J])*C1*RP[J-1]*C[I-J];
      }
      C[I]=-srem(C[I],F)/I;
    }

    N=(L<DegH)?L:DegH;
    S=0;for(I=0;I<N;I++) S+=RP[I]*DH[I]*C[L-1-I];S=srem(S,F);

    K=inversePolyn(DF,F);
    InvDF=ptozp(K);
    InvDFD=sdiv(InvDF,K);

    Inv=newvect(Len);
    for(I=1;I<Len;I++){
```

```
      Inv[I]=newvect(2);
      if(I!=II){
        K=inversePolyn(D[I][0],F);
        Inv[I][0]=ptozp(K);
        Inv[I][1]=sdiv(Inv[I][0],K);
      }
    }

    InvA=1;InvAD=1;
    for(I=1;I<Len;I++)
      if(I!=II){
        A=srem(InvA*Inv[I][0],F);
        InvA=ptozp(A);
        InvAD*=Inv[I][1]*sdiv(InvA,A);
      }

    InvG=1;InvGD=1;
    for(I=1;I<Len;I++)
      if(I!=II){
        GP=rs(Inv[I][0],D[I][1],F);
        InvG*=GP[0];
        InvG=srem(InvG,F);
        InvGD*=Inv[I][1]^D[I][1]*GP[1];
      }

    AP=rs(InvA,L-1,F);
    DFP=rs(InvDF,2*L-1,F);

    UN=srem(DFP[0]*AP[0]*InvG,F);
    UD=fac(L-1)*InvDFD^(2*L-1)*DFP[1]*InvAD^(L-1)*AP[1]*InvGD;

    RN=srem(S*UN,F);RD=UD;
    Res=[RD,RN]; /*Res=RN/RD;*/

    print(rtostr(F)+" : "+rtostr(Res));
  }
}

/* 剰余体上での多項式の冪乗計算（繰り返し２乗法 + 整数係数化処理）*/
def rs(A,LP,F){
  K=1;PN=1;ZD=1;
  while(1){
    if(LP%2==1){P=srem(PN*A,F);PN=ptozp(P);ZD*=sdiv(PN,P)*K;}
    LP=idiv(LP,2);if(LP==0) break;
    B=srem(A*A,F);A=ptozp(B);K=sdiv(A,B)*K*K;
  }
  return [PN,ZD];
}

/* 逆元計算 */
/* Asir のライブラリファイル sp を使うので load しておく．*/
def inversePolyn(Poly,F){
  X=var(F);
  Alg=newalg(F);
  Inv=simpalg(1/subst(Poly,X,Alg));
  InvR=algtorat(Inv);
  if(var(InvR)!=0) InvR=subst(InvR,var(InvR),X);
  return InvR;
}
end$
```