数理解析研究所講究録
第1907巻 2014年 71-79

安定化手法の発展形ISCZ法における シンボルリストの新しい評価法

伊井 誠和*
東邦大学大学院 理学研究科
Tomokazu Ii
Graduate School of Science, Toho University

白柳 潔†
東邦大学 理学部
Kiyoshi Shirayanagi
Faculty of Science, Toho University

1 はじめに

安定化理論 [1, 2] は、近似計算で実行すると不安定となるアルゴリズムに対し、それを変形して近似計算で実行しても誤差の影響を抑制し、安定な出力が得られるようにするための理論である。本論では、その安定化手法の発展形であるISCZ法 [3, 4, 5] で用いるシンボルリストの評価方法について提案する。2節で、安定化理論とISCZ法について復習する。3節で、提案するシンボルリストの評価方法を、4節で、この評価方法をISCZ法のユークリッドの互除法アルゴリズムに適用した実験結果を、計算時間とメモリ使用量について分析し、本手法の有効性について検討する。

2 復習

2.1 安定化理論

次のアルゴリズムを対象に安定化理論の復習を簡単に行う。詳細はを参照されたい。

- データは、すべて多項式環 $R[x_1, \cdots, x_m]$ の元からなる。R は実数体の部分体である。
- データ間の演算は、$R[x_1, \cdots, x_m]$ 内の加減乗である。
- データ上の述語は、不連続点をもつとすればそれは0のみである。

*6513001i@is.sci.toho-u.ac.jp
†kiyoshi.shirayanagi@is.sci.toho-u.ac.jp

述語の不連続点が 0 という意味は、If "$C = 0$" then ... else ... のように、値が 0 か否かによって分岐が別れることである。従って、$C = 0$ の代わりに $C > 0$ や $C \leq 0$ などでもよい。上記クラスのアルゴリズムを、不連続点 0 の代数的アルゴリズムと呼ぶ。ほとんどの数式処理のアルゴリズムはこのクラスに入るか、このクラスのアルゴリズムに変換可能である。

さて、安定化の 3 つのポイントは、

- アルゴリズムの構造は変えない。
- データ領域において、ふつうの係数を区間係数に変える。
- 述語の評価の直前で、区間係数のゼロ書換えを行なう。

である。すなわち、安定化されたアルゴリズムは次のようになる。

区間領域 データ領域は区間係数多項式の集合。区間係数は $[A, B]$ なる形で、$A, B \in \mathbb{R}$, $[A, B]$ は集合 $\{r \in \mathbb{R} \mid A \leq r \leq B\}$ を意味する。

区間演算 二項演算 $\star \in \{+, -, \times, \div\}$ に対し、

$$[A, B] \star [C, D] = [min(A \star C, A \star D, B \star C, B \star D), max(A \star C, A \star D, B \star C, B \star D)]$$

ゼロ書換え 不連続点 0 をもつ述語を評価する直前で、各区間係数 $[A, B]$ に対し、

$A \leq 0 \leq B$ ならば $[A, B]$ を $[0, 0]$ に書き換えよ。
そうでないならばそのままとせよ。

今、入力 $f \in R[x_1, \cdots, x_m]$ を

$$f = \sum_{i_1, \cdots, i_m} r_{i1 \cdots im} x_1^{i_1} \cdots x_m^{i_m}$$

と表したとき、f に対する近似列 $Int(f)_{j_j}$ を

$$Int(f)_j = \sum_{i_1, \cdots, i_m} [(a_{i1 \cdots im})_j, (b_{i1 \cdots im})_j] x_1^{i_1} \cdots x_m^{i_m}$$

で定義する。ここに、すべての $i_1, \cdots, i_m$ について、

$$(a_{i1 \cdots im})_j \leq r_{i1 \cdots im} \leq (b_{i1 \cdots im})_j \ for \ \forall j$$

$$(b_{i1 \cdots im})_j - (a_{i1 \cdots im})_j \to 0 \ as \ j \to \infty$$

このとき、単に

$$Int(f)_j \to f$$

と書く。

さて、A を安定化したアルゴリズムを $Stab(\mathcal{A})$ と書くと、次が安定化理論の基本定理 [2] である。

定理 1 (安定化理論の基本定理) A は不連続点 0 の代数的アルゴリズムで、入力 $f \in R[x_1, \cdots, x_m]$ に対し正常終了するとせよ。このとき、f に対する任意の近似列 $\{Int(f)_j\}_j$ に対し、ある n が存在して、$j \geq n$ ならば、$Stab(\mathcal{A})$ は $\{Int(f)_j\}_j$ に対し正常終了し、

$$Stab(\mathcal{A})(Int(f)_j) \to \mathcal{A}(f)$$

簡明を期すため、入力は一つだけの多項式にしているが、入力はもちろん、多項式の有限集合でもよい。

2.2 ISCZ 法

2.2.1 シンボル付き区間

区間と形式的なシンボルを組み合わせた係数（シンボル付き区間）を導入する。区間は、従来と同様の意味の区間である。シンボルは、アルゴリズム実行中に現れる係数の log（記録）を取るのに使われる。例えば、入力係数が 1/3 と 1/7 だったとしよう。これらの精度 3 の区間はそれぞれ [.333, .334] と [.142, .143] である。さて、1/3 に対するシンボルを s、1/7 に対するシンボルを t として、区間と組み合わせると、それぞれ、$[[0.333, 0.334], s]$ と $[[0.142, 0.143], t]$ となる。次に、これらの間の演算、

$$[[0.333, 0.334], s] + [[0.142, 0.143], t] = [[0.333, 0.334] + [0.142, 0.143], s\dot{+}t]$$

と定義する。$[0.333, 0.334] + [0.142, 0.143]$ に対しては通常の区間演算を使う。シンボル部分 $s\dot{+}t$ は再び形式的なシンボルで、加算を実施したことを記録できれば何でもよい。効率的なシンボル付けについては 2.3 節で議論する。

アルゴリズム終了後、最終的なシンボルを正確係数に復元する。先の簡単な例で言えば、もし最終的なシンボルが $s\dot{+}t$ であったとすれば、s に 1/3 を、t に 1/7 を代入し、$\dot{+}$ には加算の意味を与えて、1/3+1/7 = 10/21 と復元する。

シンボル付き区間のことを interval-symbol、あるいは単に IS と呼ぶ。

2.2.2 手続き

A を不連続点 0 の代数的アルゴリズムとする。ISCZ 法の手続きは次の通りである。

R-to-IS 各入力係数 r を $[[A, B], Symbol_r]$ に変換する。ここに、$[A, B]$ は r の予め定められた精度の区間、$Symbol_r$ は r を表すシンボル (以下、入力シンボルと呼ぶ) である。

IS 演算 IS 間の演算を次のように実行する:

$$[[A, B], s] + [[C, D], t] = [[A, B] + [C, D], s\dot{+}t]$$

$$[[A, B], s] - [[C, D], t] = [[A, B] - [C, D], s\dot{-}t]$$

$$[[A, B], s] \times [[C, D], t] = [[A, B] \times [C, D], s\dot{\times}t]$$

すなわち、区間部分については区間演算を用い、シンボル部分については加算、減算、乗算の形式的なシンボル $\dot{+}, \dot{-}, \dot{\times}$ を使って、どういう演算が行なわれたかを記録する。

正しいゼロ書換え 任意の IS$[[A, B], s]$ に対し、$A \leq 0 \leq B$ ならば、s をそれに対応する実数 $r(s)$ に復元する。もし、$r(s) = 0$ ならば、次のステップに進む。そうでなければ、精度を上げて **R-to-IS** に戻る。

IS-to-R 出力のシンボル部分の中の各入力シンボルにそれぞれ対応する入力係数を代入し、演算シンボルに演算の意味を与えて実数値に復元する。

この手法を ISCZ 法 (IS method with correct zero rewriting) と呼ぶ。

さて、ある精度 j で $Int(f)_j$ を $Stab(\mathcal{A})$ に入力したときの実行過程は、もしアルゴリズム中のすべてのゼロ書換えが正しいならば、真の出力 f を A に入力したときの実行過程と完全に一致する。ISCZ 法では、各ゼロ書換えにおいて、それが正しいかどうかを確認する。さらに、定理 1 により、すべてのゼロ書換えが正しくなる精度が存在する。従って、ISCZ 法は有限ステップで終了し、その出力の各 IS 係数のシンボルは正しい正確係数を与える。

これを定理にまとめる。

定理 2 (ISCZ 法の停止性と正当性) A が入力 I で正常終了するとせよ。このとき、A に対する *ISCZ* 法は、常に有限ステップで終了し、正しい結果、すなわち、$\mathcal{A}(I)$ の出力と同じ結果を与える。

ISCZ 法の利点は、ISZ 法と違い、出力の正当性を確認する必要がないことである。任意の IS$[[A,B],s]$ に対し、$A \leq 0 \leq B$ でない限り、正確計算をスキップすることができ、浮動小数点計算だけで済む。換言すれば、ゼロでない係数についての正確計算を省略することができる。従って、本手法は、$A \leq 0 \leq B$ でない場合が $A \leq 0 \leq B$ である場合よりも多ければ多いほど有効であるということができる。

2.3 シンボルリスト

ISCZ 法では、IS のシンボル部分が膨張してしまうことがある。それを防ぐため、IS に用いるシンボルは全て整数とし、計算の経過を行番号に対応させて保存する行列 (シンボルリスト) を用いることとする。シンボルリストについては [4] において提案されている。

シンボルリストには n 行 3 列の行列を用い、0 列目・1 列目には数値もしくはシンボル、2 列目には演算を保存する。もし、シンボルが表すものが実数であれば、0 列目に実数、1 列目と 2 列目に -1 を保存し、演算ではないことを示す。

シンボル番号は、IS 演算を行うごとに新しい番号をつけていく。

例えば

$$\begin{aligned} \frac{1}{3}+\frac{1}{7}\times\frac{2}{9} &= [[0.333,0.334],0]+[[0.142,0.143],1]\times[[0.222,0.223],2] \\ &= [[0.333,0.334],0]+[[0.031524,0.031889],3] \\ &= [[0.364524,0.365889],4] \end{aligned}$$

の計算経過をシンボルリストに保存する場合、

$$SL = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \begin{pmatrix} \frac{1}{3} & -1 & -1 \\ \frac{1}{7} & -1 & -1 \\ \frac{2}{9} & -1 & -1 \\ 1 & 2 & \times \\ 0 & 3 & + \end{pmatrix} \end{array}$$

となる。

3 シンボルリストの評価方法

3.1 行列法

2.3 節で保存したシンボルリストを用いて、シンボル番号 4 を評価する手順によって説明する。

$$SL = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \begin{pmatrix} \frac{1}{3} & -1 & -1 \\ \frac{1}{7} & -1 & -1 \\ \frac{2}{9} & -1 & -1 \\ 1 & 2 & \times \\ 0 & 3 & + \end{pmatrix} \end{array}$$

流れとしては、シンボルリストから評価行列を作り、その評価行列を用いて計算することになる。

各行の 0 列目にはその行にいくつのシンボルリストから取った 3 つ組が存在するかを保存する。例えば 0 行目には評価を行うシンボル番号 4 に対応するシンボルリストの行のみを入れるため、3 つ組の数は 1 となり、0 行 0 列目に 1 が保存される。

$$eva = \begin{matrix} \\ 0 \end{matrix} \begin{matrix} \begin{matrix} 0 & 1 & 2 & 3 \end{matrix} \\ \begin{pmatrix} 1 & 0 & 3 & + \end{pmatrix} \end{matrix}$$

eva 行列の n 行 3m+3 列目 ($m \geq 0, m \in \mathbb{Z}$) を読み、演算記号ならばシンボルリストから 3m 列目・3m+1 列目のシンボルに対応するシンボルリストの行を n+1 行目に左詰めで入れる。演算記号ではなく-1 ならば何も行わずスキップする。n 行目の全ての列が終わったら n+1 行目以降にも同様の操作を行い、n 行目に演算シンボルがなくなるまで繰り返す。

$$eva = \begin{matrix} \\ 0 \\ 1 \end{matrix} \begin{matrix} \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 \end{matrix} \\ \begin{pmatrix} 1 & 0 & 3 & + & & & \\ 2 & \frac{1}{3} & -1 & -1 & 1 & 2 & \times \end{pmatrix} \end{matrix}$$

$$eva = \begin{matrix} \\ 0 \\ 1 \\ 2 \end{matrix} \begin{matrix} \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 \end{matrix} \\ \begin{pmatrix} 1 & 0 & 3 & + & & & \\ 2 & \frac{1}{3} & -1 & -1 & 1 & 2 & \times \\ 2 & \frac{1}{7} & -1 & -1 & \frac{2}{9} & -1 & -1 \end{pmatrix} \end{matrix}$$

評価行列が完成してから演算を行う。n 行の評価行列での演算は n 行目,$n-1$ 行目,$\cdots$,0 行目の順に行い、演算シンボルがあれば 1 行下に保存されている実数を用いて計算を行うこととする。計算結果は演算シンボルが属する 3 つ組に上書きする。

$$eva = \begin{matrix} \\ 0 \\ 1 \\ 2 \end{matrix} \begin{matrix} \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 \end{matrix} \\ \begin{pmatrix} 1 & 0 & 3 & + & & & \\ 2 & \frac{1}{3} & -1 & -1 & 1 & 2 & \times \\ 2 & \frac{1}{7} & -1 & -1 & \frac{2}{9} & -1 & -1 \end{pmatrix} \end{matrix}$$

$$\downarrow$$

$$eva = \begin{matrix} \\ 0 \\ 1 \\ 2 \end{matrix} \begin{matrix} \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 \end{matrix} \\ \begin{pmatrix} 1 & 0 & 3 & + & & & \\ 2 & \frac{1}{3} & -1 & -1 & \frac{2}{63} & -1 & -1 \\ 2 & \frac{1}{7} & -1 & -1 & \frac{2}{9} & -1 & -1 \end{pmatrix} \end{matrix}$$

また、同じシンボルを複数回評価するのは無駄であるため、一度評価されたシンボルはシンボルリストに実数として上書きする。

$$SL = \begin{matrix} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{matrix} \begin{matrix} \begin{matrix} 0 & 1 & 2 \end{matrix} \\ \begin{pmatrix} \frac{1}{3} & -1 & -1 \\ \frac{1}{7} & -1 & -1 \\ \frac{2}{9} & -1 & -1 \\ 1 & 2 & \times \\ 0 & 3 & + \end{pmatrix} \end{matrix} \rightarrow SL = \begin{matrix} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{matrix} \begin{matrix} \begin{matrix} 0 & 1 & 2 \end{matrix} \\ \begin{pmatrix} \frac{1}{3} & -1 & -1 \\ \frac{1}{7} & -1 & -1 \\ \frac{2}{9} & -1 & -1 \\ \frac{2}{63} & -1 & -1 \\ 0 & 3 & + \end{pmatrix} \end{matrix}$$

この操作を繰り返し、0 行目が計算された時の 0 行 1 列目が評価結果となる。

3.2 式列法

式列法の基本形は次のようになる。

$$\begin{array}{cc} & \begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array} \\ eva = & \begin{pmatrix} \text{“(”} & \text{“}a\text{”} & s & \text{“}=\text{”} & \text{“}s\text{”} & t & \star & \text{“}s\text{”} & u & \text{“)”} \end{pmatrix} \end{array}$$

ここで、s,t,u はシンボル番号を、“a” は次の数が評価されるシンボルであることを、“s” は次の数がシンボルであることを、$\star$ は演算記号を表す。この行列を 0 列目から走査し、以下の条件に当てはまったら操作を加える。

- n 列目が “s” かつ $n+1$ 列目が表すシンボルが演算のとき
  n 列目を起点として基本形に展開する。$n+2$ 列目以降を右に 8 つシフトする。

$$\begin{array}{cc} & \begin{array}{cccccccc} \cdots & n & n+1 & n+2 & n+3 & n+4 & n+5 & \cdots \end{array} \\ eva = & \begin{pmatrix} \cdots & \text{“}s\text{”} & s & \star & \text{“}s\text{”} & t & \text{“)”} & \cdots \end{pmatrix} \end{array}$$

$$\downarrow$$

$$\begin{array}{cc} & \begin{array}{cccccccccccc} \cdots & n & n+1 & n+2 & n+3 & n+4 & n+5 & n+6 & n+7 & n+8 & n+9 & n+10 & \cdots \end{array} \\ eva = & \begin{pmatrix} \cdots & \text{“(”} & \text{“}a\text{”} & s & \text{“}=\text{”} & \text{“}s\text{”} & u & \star & \text{“}s\text{”} & v & \text{“)”} & \star & \cdots \end{pmatrix} \end{array}$$

- n 列目が “s” かつ $n+1$ 列目が表すシンボルが実数のとき
  シンボルを表す “s” を実数を表す “r” に変え、後ろに実数を入れる。

$$\begin{array}{cc} & \begin{array}{cccccccc} \cdots & n & n+1 & n+2 & n+3 & n+4 & n+5 & \cdots \end{array} \\ eva = & \begin{pmatrix} \cdots & \text{“}s\text{”} & s & \star & \text{“}s\text{”} & t & \text{“)”} & \cdots \end{pmatrix} \end{array}$$

$$\downarrow$$

$$\begin{array}{cc} & \begin{array}{cccccccc} \cdots & n & n+1 & n+2 & n+3 & n+4 & n+5 & \cdots \end{array} \\ eva = & \begin{pmatrix} \cdots & \text{“}r\text{”} & p & \star & \text{“}s\text{”} & t & \text{“)”} & \cdots \end{pmatrix} \end{array}$$

- $n+4$ 列目と $n+7$ 列目が共に “r” のとき
  演算を行い、$n+10$ 列目以降を左に 8 つシフトする。

$$\begin{array}{cc} & \begin{array}{cccccccccccc} \cdots & n & n+1 & n+2 & n+3 & n+4 & n+5 & n+6 & n+7 & n+8 & n+9 & n+10 & \cdots \end{array} \\ eva = & \begin{pmatrix} \cdots & \text{“(”} & \text{“}a\text{”} & s & \text{“}=\text{”} & \text{“}r\text{”} & p & \star & \text{“}r\text{”} & q & \text{“)”} & \star & \cdots \end{pmatrix} \end{array}$$

$$\downarrow$$

$$\begin{array}{cc} & \begin{array}{cccc} \cdots & n & n+1 & n+2 & \cdots \end{array} \\ eva = & \begin{pmatrix} \cdots & \text{“}r\text{”} & r & \star & \cdots \end{pmatrix} \end{array}$$

また、演算後は行列法と同様にシンボルリストにも演算結果を上書きする。

この操作を繰り返し、0 列目が “r” になった時の 1 列目が評価結果となる。

4 ユークリッドの互除法への適用

前述のシンボルリストの評価方法をユークリッドの互除法に適用し実験を行った。

表 1: 実験結果 (安定化計算・有理係数)

入出力次数	SP	ZR	SL	行列法			式列法		
				Time	Memory	Elements	Time	Memory	Elements
5-3-2	3	7	51	0	208.51k	102	0	139.71k	42
9-7-3	8	51	165	47	2.67M	204	31	2.92M	106
10-8-6	4	25	141	15	0.90M	178	31	1.03M	82
39-26-19	9	341	1495	436	30.83M	4778	453	28.98M	234

4.1 実験方法

- 1 変数多項式のユークリッドの互除法を行う。
- 以下の 4 つの方法で計算開始から出力までの時間・メモリ使用量を計測する。ISCZ 法については評価行列の最大要素数も計測する。
  1. ISCZ 法のユークリッドの互除法アルゴリズムに、行列法の評価方法を組み込んだもの
  2. ISCZ 法のユークリッドの互除法アルゴリズムに、式列法の評価方法を組み込んだもの
  3. 自作の正確演算でのユークリッドの互除法アルゴリズム
  4. Maple に組み込まれている GCD 関数 (正確演算)
- 実験環境は以下の通り。
  使用ソフト:数式処理ソフト Maple14
  使用コンピュータ
  OS:Windows 7 Professional
  CPU:Intel(R) Pentium(R) CPU G840 @ 2.80GHz
  メモリ:4.00GB

4.2 実験結果

実験結果表中の略号は以下の通り。
入出力次数:(入力した多項式 A の次数)-(入力した多項式 B の次数)-(出力された多項式の次数)
SP:開始精度から 1 桁ずつ増やした時の出力精度 (開始精度は有理係数では 3 桁、無理係数では 1 桁)
ZR:ゼロ書き換え回数
SL:シンボルリストの長さ
Time:ミリ秒
Memory:k=kib M=Mib
Elements:評価行列の要素数

表 1 と表 2 は入力多項式の係数が有理数だけのものである。有理係数での実験では行列法と式列法についてはどちらが優位とは決められなかった。

また、自作の正確演算アルゴリズム、Maple に組み込まれている GCD 関数と比べると時間・メモリ使用量共に ISCZ 法を用いない方が優位であった。

表 2: 実験結果 (厳密計算・有理係数)

入出力次数	SP	ZR	SL	自作関数 (正確演算)		Maple 関数 (正確演算)	
				Time	Memory	Time	Memory
5-3-2	3	7	51	0	8.43k	0	10.94k
9-7-3	8	51	165	0	35.31k	0	12.09k
10-8-6	4	25	141	0	18.46k	0	12.52k
39-26-19	9	341	1495	0	164.66k	0	28.18k

表 3: 実験結果 (安定化計算・無理係数)

入出力次数	SP	ZR	SL	行列法			式列法		
				Time	Memory	Elements	Time	Memory	Elements
3-2-1	2	9	31	312	15.88M	77	94	9.84M	42
7-4-3	3	21	75	265	13.37M	76	203	9.70M	42
11-6-3	5	41	175	1607	86.76M	178	1216	69.57M	82
15-11-5	11	133	369	8658	472.43M	306	5584	297.22M	146

表 4: 実験結果 (厳密計算・無理係数)

入出力次数	SP	ZR	SL	自作関数 (正確演算)		Maple 関数 (正確演算)	
				Time	Memory	Time	Memory
3-2-1	2	9	31	31	1.41M	0	1.68M
7-4-3	3	21	75	63	1.29M	47	1.46M
11-6-3	5	41	175	0	0.76M	62	4.48M
15-11-5	11	133	369	46	4.76M	390	25.13M

表3と表4は入力多項式の係数に無理数が含まれるものである。無理係数での実験では行列法よりも式列法の方が優位であった。

しかし、やはり自作の正確演算アルゴリズム、Mapleに組み込まれているGCD関数と比べると、時間・メモリ使用量共にISCZ法を用いない方が優位であった。

このことから、無理係数に限っては式列法が有効であると考えられる。また、ISCZ法自体がユークリッドの互除法には効果が薄いということもわかった。

5 おわりに

ISCZ法のシンボル部分膨張に対する解決策として提案されていたシンボルリストの評価の方式について提案した。今回のユークリッドの互除法では、無理係数において式列法の方が優位ではないかという予想ができる結果となったが、今後は他の不安定なアルゴリズムでの実験が必要となる。

謝辞

本研究および発表を行うにあたり、東京理科大学の関川浩教授より多くの助言を頂きました。この場をお借りして感謝申し上げます。

[1] 白柳 潔: 『コンピュータのカオスをおさえる-新しい「安定」計算術』(コミュニケーションサイエンスシリーズ 1), NTT コミュニケーション科学基礎研究所監修, NTT 出版株式会社, 東京, 2003

[2] 白柳 潔: 代数的アルゴリズムの安定化理論, コンピュータソフトウェア, **Vol.19 No.3**, 2002, 49–65

[3] 白柳 潔, 関川 浩: 安定化理論に基づく ISCZ 法の凸包構成への応用, 京都大学数理解析研究所講究録 **1815**, 2012, 133–142

[4] 白柳 潔, 関川 浩: 安定化理論に基づく ISCZ 法の有効性について, 京都大学数理解析研究所講究録 **1814**, 2012, 29–35

[5] 白柳 潔, 関川 浩: 安定化理論に基づく log method について, 京都大学数理解析研究所講究録 **1666**, 2009, 98–105