①

報告番号 甲第 3802 号

多項式剰余列の安定な生成法とその応用に関する研究

大 迫 尚 行

目 次

第 1 章

序論

古典的な Euclid 算法は，整数論において，2 つの整数の最大公約数を求める計算法，Diophantus 方程式の解法あるいは実数を連分数展開する方法としてよく知られている [1. 35]．整数論に限らず，Euclid 算法の応用分野は広く，C. Brezinski によって，6000 を越える文献が報告され，代表的に

- 物理，化学
- 信号処理
- 天文学
- Computer Algebra（計算機代数）
- 確率過程，統計学
- 数値解析
- 整数論
- 電子工学

が挙げられている [4]．

2 つの整数を 2 つの多項式 F，G で置き換えれば，Euclid 算法は，多項式剰余列を生成し，その中の最低次数多項式が最大公約因子となる．多項式剰余列の任意の要素 P は 2 つの多項式 A，B によって

$$\begin{aligned} P &= AF + BG, \\ \deg A &< \deg G - \deg P, \\ \deg B &< \deg F - \deg P \end{aligned}$$

と一意的に書ける．剰余多項式 P と同時に多項式 A，B を求める算法は，拡張 Euclid 算法と呼ばれる．この拡張 Euclid 算法は，非線形の関数近似，代表的に関数の Padé 近似，

有理関数補間，連分数展開の計算，またDiophantus方程式の多項式版として，孫子剰余定理の応用（代数方程式の数値解法）に用いられる [25]．特に，拡張Euclid算法によって得られるPadé近似は，偏微分方程式のA-stable法による数値解法あるいは特殊関数の数値計算法と関連していることが20世紀半ばの物理学者によって発見され，この頃からPadé近似の有用性が見直されてきた．最近，流体力学における基礎方程式（Navier-Stokes方程式，Eulerの運動方程式，Prandtlの境界層方程式）に対して，Padé近似による非線形の関数近似の手法が解法に取り入れられている [23]．非線形方程式の解法において，よく用いられる手法として関数を線形化して解くNewton法があるが，非線形項が線形化して解いた近似解に大きく影響する問題に対しては，非線形近似による方法が有効となってくる．

拡張Euclid算法の応用としては，他に解析関数の零点あるいは，極を求める問題，ディジタル信号処理における数値積分変換（Fourier変換，Laplace変換），インパルス応答から伝達関数を求めるZ変換 [22]，制御工学におけるシステム同定問題 [17]，数値等角写像などがある．

多項式を対象としたEuclid算法には，次の克服すべき問題点がある．有理係数の多項式を対象にすれば，数式処理では，計算の中間結果が莫大になる中間膨張と呼ばれる現象によって，計算量が多くなる [32]．浮動小数点演算の下では，丸め誤差の影響を受けやすく，桁溢れの危険性も大きい．近年，数式処理の立場から代数演算と数値的な近似演算とを融合した近似代数の名の下で，近似GCDの計算とその応用に関する研究が精力的に行われている [7, 28, 31]．一方，数値計算の立場では，実係数または複素係数の多項式の剰余列生成において，計算量が少なくて，かつ数値的に安定な計算法について関心が高まっている [10, 11, 19, 30]．しかしながら，安定性の観点からいえば，未だ決定的な方法はないといえる．

我々は，数値計算の立場から多項式剰余列の安定な生成法について研究し，多項式の係数に関する二乗ノルムの意味で決定的に安定な算法を得た．本論文の構成は次のように構成されている．

まず多項式に関する記号と定義および多項式剰余列に関する基本的な定理を第2章で提示し，第3章では，G.E.Collinsが導入した部分終結式を用いた多項式剰余列の安定な生成法およびその拡張算法について述べる．一般に用いられるEuclid算法が数値的に不安定になる原因を明らかにする．第4章では，Givens回転を用いた多項式剰余列の安定な生成法を示す．2つの多項式 F，G によって定義される線形空間 $L_k(F,G)$ または，その部分空間を行列 L_k およびその部分行列に対応させ，それらをQR分解によって上三角化することが算法の基本となっている．この結果は，直ちに多項式剰余列の拡張算法に応用される．この拡張算法によって得られる多項式系の性質について触れる．本章の終りで，Givens回転による算法の安定性を検証する．第5章では，上述の安定な算法を関数の有理補間へ応用する．複素領域における有理補間の誤差評価およびPadé近似の事後誤差評価について考察する．付録として，Fortan 90による本算法に関するプログラムを掲載する．

第2章

基本的事項

2.1 多項式に関する記号および定義

まず最初に，多項式に関する術語と記号を導入する．本論では，1変数 z の K 上の多項式環を取り扱い，K は，実数あるいは複素数全体とする．記号で $K[z]$ と書く．原則として，多項式は大文字，変数および係数は小文字で表すことにする．多項式 $F, G \in K[z]$ を

$$F = f_m z^m + f_{m-1} z^{m-1} + \ldots + f_0 \tag{2.1}$$

$$G = g_n z^n + g_{n-1} z^{n-1} + \cdots + g_0,\ f_m \neq 0,\ g_n \neq 0,\ n \leq m \tag{2.2}$$

とする．

$\mathrm{lc}(F)$	F の主係数 (leading coefficient) f_m．
$\deg F$	F の次数 m．
F の形式的次数	必ずしも $f_m \neq 0$ でないとき，m を F の形式的次数と呼ぶ．
$F \bmod G$	F を G で割った剰余多項式．
$G \mid F$	F が G で割り切れるとき，G を F の因子といい，$G \mid F$ で表す．
$\gcd(F,G)$	F と G の最大公約因子．特に，$\gcd(F,G)=1$ のとき，F と G は互いに素であるという．
$\sim$	F，G が定数倍を除いて等しいとき，F と G は互いに相似であるといい，$F \sim G$ で表す．

2.2 Euclid 算法と多項式剰余列

式 (2.1), (2.2) で与えられる 2 つの多項式 F, G の多項式剰余列 $\{P_i\}_{i=-1}^{t}$ を次の Euclid 算法によって定義する．$P_{-1}=F$, $P_0=G$ とおき

$$\begin{aligned} &P_{i+1}=P_{i-1} \bmod P_i,\ i=0,1,\ldots,t \\ &P_{t+1}=0,\ P_t\neq 0. \end{aligned} \tag{2.3}$$

特に，$P_t=\gcd(F,G)$ である．

P_0 以降の多項式が 1 次ずつ減少するとき，すなわち

$$\deg P_{i-1}-\deg P_i=1,\ i=1,2,\ldots,t$$

のとき，多項式剰余列は正規 (normal) であるといい，そうでないとき，不正規 (abnormal) であるという．式 (2.3) において，P_{i-1} を P_i で割った商多項式を Q_{i+1} をすれば

$$P_{i+1}=P_{i-1}-Q_{i+1}P_i,\ \deg P_{i+1}<\deg P_i \tag{2.4}$$

と書ける．この漸化式を用いて，剰余列要素 P_i に 2 つの多項式 A_i, B_i を付加した (P_i,A_i,B_i) を計算する．

$$\begin{aligned} (P_{-1},A_{-1},B_{-1}) &= (F,1,0), \\ (P_0,A_0,B_0) &= (G,0,1), \\ (P_{i+1},A_{i+1},B_{i+1}) &= (P_{i-1},A_{i-1},B_{i-1})-Q_{i+1}\cdot(P_i,A_i,B_i),\ i=0,1,\ldots,t \end{aligned} \tag{2.5}$$

この計算法は，F と G の多項式剰余列 $\{P_i\}_{i=-1}^{t}$ を計算する Euclid 算法を拡張したものであるので，拡張 Euclid 算法と呼ばれる．拡張 Euclid 算法で得られる多項式列は次の性質をもつ．

補助定理 2.2.1 拡張 Euclid 算法で定義された $\{P_i,A_i,B_i\}_{i=-1}^{t}$ において，多項式 A_i, B_i の次数について

$$\deg A_i = \deg G-\deg P_{i-1}, \tag{2.6}$$

$$\deg B_i = \deg F-\deg P_{i-1},\ 1\le i\le t \tag{2.7}$$

が成り立つ．

[証明]　添字 i に関する帰納法で示す．

i) $i=1$ のとき，(2.5) より

$$\begin{aligned} A_1 &= A_{-1}-Q_1A_0 \\ &= 1, \end{aligned} \tag{2.8}$$

$$\begin{aligned} B_1 &= B_{-1}-Q_1B_0 \\ &= -Q_1 \end{aligned} \tag{2.9}$$

である．ここで，多項式 Q_1 は P_{-1} を P_0 で割った商多項式，すなわち F を G で割った商多項式であるので，$\deg Q_1=\deg F-\deg G$．また，$P_0=G$ より

$$\begin{aligned} \deg A_1 &= \deg G-\deg P_0, \\ \deg B_1 &= \deg F-\deg P_0 \end{aligned}$$

となり，$i=1$ のときは成立する．

ii) 帰納法の仮定として，$1\le j\le i$ について成り立っているとする．このとき，$i+1$ についても成立することを示す．(2.5) 式より A_{i+1}, B_{i+1} はそれぞれ

$$A_{i+1} = A_{i-1}-Q_{i+1}A_i \tag{2.10}$$

$$B_{i+1} = B_{i-1}-Q_{i+1}B_i \tag{2.11}$$

で与えられる．ここで，帰納法の仮定および $\deg P_i$ の i に関する単調減少性より

$$\begin{aligned} \deg A_{i-1} &= \deg G-\deg P_{i-2} \\ &< \deg G-\deg P_{i-1}=\deg A_i \end{aligned}$$

となる．同様に

$$\deg B_{i-1}<\deg B_i.$$

これらの評価式と (2.10), (2.11) より A_{i+1}, B_{i+1} の次数はそれぞれ

$$\begin{aligned} \deg A_{i+1} &= \deg Q_{i+1}+\deg A_i \\ \deg B_{i+1} &= \deg Q_{i+1}+\deg B_i \end{aligned}$$

となる．上式において，Q_{i+1} は P_{i-1} を P_i で割った商多項式であるので

$$\deg Q_{i+1}=\deg P_{i-1}-\deg P_i$$

となる．この関係式と $\deg A_i$, $\deg B_i$ に関する帰納法の仮定より

$$\begin{aligned} \deg A_{i+1} &= (\deg P_{i-1}-\deg P_i)+(\deg G-\deg P_{i-1}) \\ &= \deg G-\deg P_i \\ \deg B_{i+1} &= (\deg P_{i-1}-\deg P_i)+(\deg F-\deg P_{i-1}) \\ &= \deg F-\deg P_i \end{aligned}$$

となり，$i+1$ についても成立する．

よって，i), ii) より (2.6), (2.7) が示された．■

定理 2.2.1 [31] 拡張 Euclid 算法で定義された $\{P_i, A_i, B_i\}_{i=-1}^{t}$ において，番号 i $(1 \leq i \leq t)$ について

$$P_i = A_i F + B_i G, \tag{2.12}$$
$$\deg A_i < \deg G - \deg P_i,\ \deg B_i < \deg F - \deg P_i \tag{2.13}$$

が成立する．

[証明]　式 (2.13) については，補助定理 2.2.1 の (2.6) 式より

$$\begin{aligned} \deg A_i &= \deg G - \deg P_{i-1} \\ &< \deg G - \deg P_i \end{aligned}$$

となる．$\deg B_i$ についても同様にして示される．
式 (2.12) について，i に関する帰納法で示す．

i) $i=1$ のとき，補助定理 2.2.1 の (2.8)，(2.9) 式より $A_1 = 1$，$B_1 = -Q_1$ であるので

$$\begin{aligned} P_1 &= P_{-1} - Q_1 P_0 \\ &= 1 \cdot F + (-Q_1) \cdot G \\ &= A_1 F + B_1 G \end{aligned}$$

となる．よって，$i=1$ のときは成立する．

ii) 帰納法の仮定として，$1 \leq j \leq i$ について成り立っているとする．このとき，$i+1$ についても成立することを示す．(2.5) 式より

$$\begin{aligned} P_{i+1} &= P_{i-1} - Q_{i+1} P_i \\ &= (A_{i-1} F + B_{i-1} G) - Q_{i+1}(A_i F + B_i G) \\ &= (A_{i-1} - Q_{i+1} A_i) F + (B_{i-1} - Q_{i+1} B_i) G \\ &= A_{i+1} F + B_{i+1} G \end{aligned}$$

となる．上式において，2 番目の等式は帰納法の仮定を用いた．よって，$i+1$ についても成立する．

よって，i)，ii) より (2.12) 式が示された．■
この定理は，剰余列の生成法によらないことが次の定理でわかる [[29], pp.14–15].

定理 2.2.2 [32] 式 (2.1)，(2.2) で与えられる 2 つの多項式の多項式剰余列において，任意の剰余列要素 P_i $(1 \leq i \leq t)$ に対して，次数に関する条件

$$\begin{aligned} &\deg A_i < \deg G - \deg P_i \\ &\text{または} \deg B_i < \deg F - \deg P_i \end{aligned} \tag{2.14}$$

の下で

$$P_i = A_i F + B_i G \tag{2.15}$$

を満たす多項式 A_i，B_i は一意に存在する．

[証明]　剰余列要素 P_i に対して，定理の条件を満たす多項式 A_i，B_i の存在は定理 2.2.1 よりいえるので，一意性を示す．上の条件を満たす 2 つの多項式が他にも存在したとしてそれらを $\tilde{A}_i$，$\tilde{B}_i$ とする．

$$\deg \tilde{A}_i < \deg G - \deg P_i \tag{2.16}$$
$$P_i = \tilde{A}_i F + \tilde{B}_i G \tag{2.17}$$

(2.15) 式と (2.17) 式の両辺差し引けば

$$0 = (A_i - \tilde{A}_i) F + (B_i - \tilde{B}_i) G$$

上式において，$\bar{F} = F/\gcd(F,G)$，$\bar{G} = G/\gcd(F,G)$ とおき，両辺を $\gcd(F,G)$ で割れば

$$0 = (A_i - \tilde{A}_i) \bar{F} + (B_i - \tilde{B}_i) \bar{G} \tag{2.18}$$

となる．ここで，$\bar{F}$ と $\bar{G}$ は互いに素であるので

$$\bar{G} | (A_i - \tilde{A}_i). \tag{2.19}$$

一方，次数に関する条件式 (2.14)，(2.16) より

$$\begin{aligned} \deg(A_i - \tilde{A}_i) &< \deg G - \deg P_i \\ &= (\deg \bar{G} + \deg \gcd(F,G)) - \deg P_i \\ &= \deg \bar{G} - (\deg P_i - \deg \gcd(F,G)) \leq \deg \bar{G} \end{aligned} \tag{2.20}$$

である．したがって (2.19) と (2.20) より $A_i - \tilde{A}_i = 0$ すなわち $\tilde{A}_i = A_i$ となる．さらに，(2.18) 式より $\tilde{B}_i = B_i$ となり，一意性が示された．多項式 B_i に関する次数条件

$$\deg B_i < \deg F - \deg P_i$$

の場合も同様にして示される．■

第 3 章

部分終結式による多項式剰余列の安定な生成法

まず，この節の算法に必要な多項式に関する術語と演算を新たに導入する．

$\|F\|_\infty$　多項式 F の無限大ノルムと呼び，係数の絶対値の最大値で定義する．

枢軸 (pivot)　同じ次数の多項式列 $\{G_i\}_{i=0}^k$ において，G_{i_0} を用いて，$G_i \bmod G_{i_0}$ によって，各 G_i $(i \neq i_0)$ の主係数を消去するとき，G_{i_0} の主係数を枢軸 (pivot) といい，G_{i_0} を枢軸多項式という．

$\mathrm{Piv}\{G_i\}_{i=0}^k$　同じ次数の多項式列 $\{G_i\}_{i=0}^k$ に対して，その中で主係数の絶対値最大の多項式と G_0 とを交換する操作．この操作を枢軸選び (pivoting) という．

$\mathrm{Elim}\{G_0, G_1\}$　同じ次数の多項式 G_0，G_1 に対して，次で定義される主係数消去の演算．

$$\mathrm{Elim}\{G_0, G_1\} := \begin{cases} G_1 - \frac{\mathrm{lc}(G_1)}{\mathrm{lc}(G_0)} G_0, & |\mathrm{lc}(G_0)| > |\mathrm{lc}(G_1)| \\ G_0 - \frac{\mathrm{lc}(G_0)}{\mathrm{lc}(G_1)} G_1, & |\mathrm{lc}(G_0)| \leq |\mathrm{lc}(G_1)|. \end{cases}$$

上式右辺の第二項の乗数の絶対値は 1 以下であることを注意しておく．

3.1　部分終結式

2 つの多項式

$$\begin{aligned} F &= f_m z^m + f_{m-1} z^{m-1} + \cdots + f_0, \\ G &= g_n z^n + g_{n-1} z^{n-1} + \cdots + g_0, \ f_m \neq 0, \ g_n \neq 0, \ n \leq m \end{aligned}$$

に対して，次の $(n-j)+(m-j)$ 個の多項式系

$$F, zF, \ldots, z^{n-j-1}F, G, zG, \ldots, z^{m-j-1}G$$

の任意の線形結合

$$R = \sum_{r=0}^{n-j-1} a_r z^r F + \sum_{s=0}^{m-j-1} b_s z^s G \tag{3.1}$$

を考える．ここで，番号 j は，$0 \leq j < n$ なる整数である．多項式 R の形式的次数は $m+n-j-1$ 次であるので

$$R = \sum_{r=0}^{m+n-j-1} r_r z^r$$

とすれば，式 (3.1) の左辺は

$$R = \left[z^{m+n-j-1}, z^{m+n-j-2}, \ldots, z, 1\right] \begin{bmatrix} r_{m+n-j-1} \\ r_{m+n-j-2} \\ \vdots \\ r_1 \\ r_0 \end{bmatrix}$$

と書ける．同様にして右辺は

$$\left[z^{m+n-j-1}, z^{m+n-j-2}, \ldots, z, 1\right] \left\{ a_{n-j-1} \begin{bmatrix} f_m \\ f_{m-1} \\ \vdots \\ f_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} + \cdots + a_0 \begin{bmatrix} 0 \\ \vdots \\ 0 \\ f_m \\ \vdots \\ f_1 \\ f_0 \end{bmatrix} \right.$$
$$\left. + b_{m-j-1} \begin{bmatrix} g_n \\ g_{n-1} \\ \vdots \\ g_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} + \cdots + b_0 \begin{bmatrix} 0 \\ \vdots \\ 0 \\ g_n \\ \vdots \\ g_1 \\ g_0 \end{bmatrix} \right\}$$

と書ける．式 (3.1) は，z に関する恒等式であるので，$m+n-j$ 次元のベクトル空間において

$$\begin{bmatrix} r_{m+n-j-1} \\ r_{m+n-j-2} \\ \vdots \\ r_1 \\ r_0 \end{bmatrix} = a_{n-j-1} \begin{bmatrix} f_m \\ f_{m-1} \\ \vdots \\ f_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} + \cdots + a_0 \begin{bmatrix} 0 \\ \vdots \\ 0 \\ f_m \\ \vdots \\ f_1 \\ f_0 \end{bmatrix}$$

$$+ b_{m-j-1} \begin{bmatrix} g_n \\ g_{n-1} \\ \vdots \\ g_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} + \cdots + b_0 \begin{bmatrix} 0 \\ \vdots \\ 0 \\ g_n \\ \vdots \\ g_1 \\ g_0 \end{bmatrix}$$

が成立する．任意の $(n-j)+(m-j)$ 個のベクトル系が線形従属であるようなベクトル空間の最大次元は，$(n-j)+(m-j)-1$ であり，この次元数が，式 (3.1) で与えられる R の係数を零にすることができる最大の個数と対応する．この事実より，多項式 R の係数が，高次から順に $(n-j)+(m-j)-1$ 個零となるような自明でない線形結合が存在する．すなわち

$$^{\exists}(a_{n-j-1}, a_{n-j-2}, \ldots, a_0, b_{m-j-1}, b_{m-j-2}, \ldots, b_0) \neq \mathbf{o} \text{ s.t.}$$

$$^{(n-j)+(m-j)-1}\left\{ \begin{bmatrix} 0 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ 0 \\ R \end{bmatrix} \right. = a_{n-j-1} \begin{bmatrix} f_m \\ f_{m-1} \\ \vdots \\ \vdots \\ f_{2j+3-n} \\ f_{2j+2-n} \\ z^{n-j-1}F \end{bmatrix} + \cdots + a_0 \begin{bmatrix} 0 \\ \vdots \\ 0 \\ f_m \\ \vdots \\ f_{j+1} \\ F \end{bmatrix}$$

$$+ b_{m-j-1} \begin{bmatrix} g_n \\ g_{n-1} \\ \vdots \\ \vdots \\ g_{2j+3-m} \\ g_{2j+2-m} \\ z^{m-j-1}G \end{bmatrix} + \cdots + b_0 \begin{bmatrix} 0 \\ \vdots \\ 0 \\ g_n \\ \vdots \\ g_{j+1} \\ G \end{bmatrix}.$$

ただし，多項式の係数の添字において，定義された領域から外れた係数は零とする．このとき，R の次数は高々 j 次となる．上式において，$a_k \neq 0$ となる番号 k が存在する．簡単のため $k=0$ とし，一般性を失うことなく $a_0 = 1$ とする．上式を線形方程式で記述すれば

$$\begin{bmatrix} f_m & & & g_n & & \\ \vdots & \ddots & & \vdots & \ddots & \\ \vdots & & f_m & \vdots & & g_n \\ \vdots & & \vdots & \vdots & & \vdots \\ f_{2j+2-n} & \cdots & f_{j+1} & g_{2j+2-m} & \cdots & g_{j+1} \\ z^{n-j-1}F & \cdots & F-R & z^{m-j-1}G & \cdots & G \end{bmatrix} \begin{bmatrix} a_{n-j-1} \\ \vdots \\ a_0 \\ b_{m-j-1} \\ \vdots \\ b_0 \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ 0 \end{bmatrix}$$

となる．左辺の係数行列は $m+n-2j$ 次の正方行列であり，その行列式は零であるので

$$\begin{vmatrix} f_m & & 0 & g_n & & 0 \\ \vdots & \ddots & & \vdots & \ddots & \\ \vdots & & f_m & \vdots & & g_n \\ \vdots & & \vdots & \vdots & & \vdots \\ f_{2j+2-n} & \cdots & f_{j+1} & g_{2j+2-m} & \cdots & g_{j+1} \\ z^{n-j-1}F & \cdots & F-R & z^{m-j-1}G & \cdots & G \end{vmatrix} = 0$$

となる．この行列式を最後の行ベクトルについて，次のように展開する．

$$\begin{vmatrix} f_m & & 0 & g_n & & 0 \\ \vdots & \ddots & & \vdots & \ddots & \\ \vdots & & f_m & \vdots & & g_n \\ \vdots & & \vdots & \vdots & & \vdots \\ f_{2j+2-n} & \cdots & f_{j+1} & g_{2j+2-m} & \cdots & g_{j+1} \\ z^{n-j-1}F & \cdots & F & z^{m-j-1}G & \cdots & G \end{vmatrix}$$

$$= \begin{vmatrix} f_m & & 0 & g_n & & 0 \\ \vdots & \ddots & & \vdots & \ddots & \\ \vdots & & f_m & \vdots & & g_n \\ \vdots & & \vdots & \vdots & & \vdots \\ f_{2j+2-n} & \cdots & f_{j+1} & g_{2j+2-m} & \cdots & g_{j+1} \\ 0 & \cdots & 1 & 0 & \cdots & 0 \end{vmatrix} \cdot R$$

上式左辺の行列式は，右辺の式より高々 j 次の多項式である．この行列式が，部分終結式であり，1967 年，G.E.Collins によって導入された [6].

定義 3.1.1 (部分終結式) [5, 6] 2 つの多項式

$$\begin{aligned} F &= f_m z^m + f_{m-1} z^{m-1} + \cdots + f_0, \\ G &= g_n z^n + g_{n-1} z^{n-1} + \cdots + g_0, \ f_m \neq 0, \ g_n \neq 0, \ n \leq m \end{aligned}$$

F と G の部分終結式 $S_j(F,G)$ を次のように定義する．

$$S_j(F,G) := \begin{vmatrix} f_m & & 0 & g_n & & 0 \\ \vdots & \ddots & & \vdots & \ddots & \\ \vdots & & f_m & \vdots & & g_n \\ \vdots & & \vdots & \vdots & & \vdots \\ f_{2j+2-n} & \cdots & f_{j+1} & g_{2j+2-m} & \cdots & g_{j+1} \\ z^{n-j-1}F & \cdots & F & z^{m-j-1}G & \cdots & G \end{vmatrix}, \ 0 \leq j \leq n-1.$$

ただし，$i < 0$ のときは，$f_i = g_i = 0$ とする．
特に，$j = 0$ のときは，単に終結式という．$S_j(F,G)$ は，高々 j 次の多項式であるので，j 次部分終結式と呼ばれる．

次に，部分終結式と多項式剰余列との関係を表す定理を挙げる．

補助定理 3.1.1 [5, 32] 2 つの多項式

$$\begin{aligned} F &= f_m z^m + f_{m-1} z^{m-1} + \cdots + f_0, \\ G &= g_n z^n + g_{n-1} z^{n-1} + \cdots + g_0, \ f_m \neq 0, \ g_n \neq 0, \ n \leq m \end{aligned}$$

において，F を G で割った商多項式を Q，剰余多項式を R とし

$$F = QG + R, \quad \deg R < \deg G$$

とする．$l = \deg R$ としたときに

$$S_j(F,G) \sim S_j(G,R), \quad 0 \leq j < l, \tag{3.2}$$

$$S_l(F,G) \sim R, \tag{3.3}$$

$$S_j(F,G) = 0, \quad l < j < n-1, \tag{3.4}$$

$$S_{n-1}(F,G) \sim R \tag{3.5}$$

が成立する．

[証明] j 次部分終結式 $S_j(F,G)$ を記号の便宜上

$$S_j(F,G) = \det[z^{n-j-1}F, \ldots, F, z^{m-j-1}G, \ldots, G]$$

と略記することにする．ここで，記号 det は行列式を表す．商多項式 Q を

$$Q = q_{m-n} z^{m-n} + q_{m-n-1} z^{m-n-1} + \cdots + q_0$$

とすれば，剰余多項式 R は

$$R = F - \sum_{r=0}^{m-n} q_r z^r G$$

と書ける．この関係式を用いれば

$$\begin{aligned} S_j(F,G) &= \det[z^{n-j-1}F, \ldots, F, z^{m-j-1}G, \ldots, G] \\ &= \det[z^{n-j-1}R, \ldots, R, z^{m-j-1}G, \ldots, G] \\ &\sim \det[z^{m-j-1}G, \ldots, G, z^{n-j-1}R, \ldots, R] \\ &= \begin{vmatrix} g_n & & 0 & & \mathbf{o}_{m-l,n-j} & \\ \vdots & \ddots & & r_l & & \\ \vdots & & g_n & \vdots & \ddots & \\ \vdots & & \vdots & \vdots & & r_l \\ \vdots & & \vdots & \vdots & & \vdots \\ g_{2j+2-m} & \cdots & g_{j+1} & r_{2j+2-n} & \cdots & r_{j+1} \\ z^{m-j-1}G & \cdots & G & z^{n-j-1}R & \cdots & R \end{vmatrix}. \end{aligned} \tag{3.6}$$

ここで，$\mathbf{o}_{m-l,n-j}$ は $m-l$ 行 $n-j$ 列の零行列である．行列式 (3.6) において，番号 j が $j \geq l$ のとき，行列式を構成する行列は下三角行列である．このとき，番号 j に応じて対角要素が零であるかないかを調べれば，(3.3)，(3.4)，(3.5) が得られる．

番号 j が $0 \leq j < l$ のとき，行列式 (3.6) の行列を次のように区分けする．

$$S_j(F,G) \sim \det \begin{bmatrix} L_{m-l} & \mathbf{o}_{m-l,n+l-2j} \\ X_{n+l-2j,m-l} & M_j(G,R) \end{bmatrix}$$

ここで，L_{m-l} は $m-l$ 次の下三角行列，$\mathbf{o}_{m-l,n+l-2j}$ は $m-l$ 行 $n+l-2j$ 列の零行列，$X_{n+l-2j,m-l}$ は $n+l-2j$ 行 $m-l$ 列の行列，$M_j(G,R)$ は $n+l-2j$ 次の正方行列で，$\det M_j(G,R) = S_j(G,R)$ である．したがって (3.2) が成立する．■

補助定理 3.1.2 [5, 32] 補助定理 3.1.1 で与えれる多項式 F，G において，F と G の多項式剰余列を

$$\begin{aligned} P_{-1} &= F, \quad P_0 = G, \\ P_i &= P_{i-2} - Q_i P_{i-1}, \quad 1 \leq i \leq t, \ P_t = \gcd(F,G), \\ & n_i = \deg P_i, \quad -1 \leq i \leq t \end{aligned}$$

で与える．ここで，Q_i は P_{i-2} を P_{i-1} で割ったときの商多項式，P_i はそのときの剰余多項式である．このとき $1 \le i \le t$ に対して

$$\begin{aligned} S_j(P_{i-2},P_{i-1}) &\sim S_j(P_{i-1},P_i), \quad 0 \le j < n_i, && (3.7)\\ S_{n_i}(P_{i-2},P_{i-1}) &\sim P_i, && (3.8)\\ S_j(P_{i-2},P_{i-1}) &= 0, \quad n_i < j < n_{i-1}-1, && (3.9)\\ S_{n_{i-1}-1}(P_{i-2},P_{i-1}) &\sim P_i && (3.10) \end{aligned}$$

が成立する．

[証明]　補助定理 3.1.1 において，F，G，Q，R をそれぞれ P_{i-2}，P_{i-1}，Q_i，P_i で置き換えれば，(3.7) ～ (3.10) が直ちに得られる．■

定理 3.1.1 [5, 6] 補助定理 3.1.2 の条件の下で，$1 \le i \le t$ に対して

$$\begin{aligned} S_j(F,G) &= 0, \quad 0 \le j < n_t, && (3.11)\\ S_{n_i}(F,G) &\sim P_i, && (3.12)\\ S_j(F,G) &= 0, \quad n_i < j < n_{i-1}-1, && (3.13)\\ S_{n_{i-1}-1}(F,G) &\sim P_i && (3.14) \end{aligned}$$

が成り立つ．

[証明]　補助定理 3.1.2 の (3.7) より，$1 \le i \le t+1$ に対して

$$\begin{aligned} S_j(F,G) &= S_j(P_{-1},P_0)\\ &\sim S_j(P_{i-2},P_{i-1}), \quad 0 \le j < n_{i-1} && (3.15) \end{aligned}$$

が成立する．上式において，$i = t+1$ とすれば

$$\begin{aligned} S_j(F,G) &\sim S_j(P_{t-2},P_{t-1})\\ &\sim S_j(P_{t-1},P_t), \quad 0 \le j < n_t \end{aligned}$$

となる．上式右辺において，$P_t = \gcd(F,G)$ であるので，P_{t-1} を P_t で割ったときの剰余多項式は零多項式となる．ここで，補助定理 3.1.1 の (3.4)，(3.5) を適用すれば

$$S_j(P_{t-1},P_t) = 0, \quad 0 \le j < n_t$$

となり，(3.11) が得られる．(3.12) ～(3.14) 式については，補助定理 3.1.2 と (3.15) 式より直ちに得られる．■

定理 3.1.1 において，多項式剰余列が正規であるとき，(3.13) の場合は起こらない．また，このとき，(3.12) と (3.14) は同一の式である．

3.2 Euclid 算法の数値的不安定性とその改良

3.2.1 基本事項

はじめに，本節以降で必要な簡単な事項を補助定理として列記しておく．

補助定理 3.2.1 多項式 F と G より生成される多項式剰余列において，その中の任意の隣接する 2 つの要素より生成される多項式剰余列は，F と G より生成される多項式剰余列の部分列である．

[証明]　Euclid 算法による多項式剰余列の定義より明らかである．■

補助定理 3.2.2 [5, 32] 多項式 F，G $(\deg F \ge \deg G)$ に対して，多項式 X，Y が

$$X \sim F, \quad Y \sim G$$

を満たすならば $S_j(X,Y) \sim S_j(F,G)$ が成り立つ．

[証明]　$m = \deg F$，$n = \deg G$ とおく．記号の便宜上，j 次部分終結式 $S_j(F,G)$ を

$$S_j(F,G) = \det[z^{n-j-1}F, \ldots, F, z^{m-j-1}G, \ldots, G]$$

と略記することにする．条件より多項式 X，Y は，零でない定数 α，β によってそれぞれ $X = \alpha F$，$Y = \beta G$ と書ける．このとき $S_j(X,Y)$ は

$$\begin{aligned} S_j(X,Y) &= \det[z^{n-j-1}X, \ldots, X, z^{m-j-1}Y, \ldots, Y]\\ &= \det[\alpha z^{n-j-1}F, \ldots, \alpha F, \beta z^{m-j-1}G, \ldots, \beta G]\\ &= \alpha^{n-j} \cdot \beta^{m-j} \cdot \det[z^{n-j-1}F, \ldots, F, z^{m-j-1}G, \ldots, G]\\ &= \alpha^{n-j} \cdot \beta^{m-j} \cdot S_j(F,G) \end{aligned}$$

となる．最後から 2 番目の等式は，行列式の線形性による．したがって $S_j(X,Y) \sim S_j(F,G)$ が成り立つ．■

この意味で，相似な多項式はしばしば同一視する．

補助定理 3.2.3 互いに相似でない同じ次数の多項式 S，T に対して，$R = S \bmod T$ とすれば

$$S \bmod R \sim T \bmod R$$

が成り立つ．

[証明]　多項式 R は

$$R = S - \alpha \cdot T, \quad \alpha \text{ は零でない定数}$$

と書ける．上式において，両辺 R で mod をとれば

$$0 = S \bmod R - \alpha \cdot T \bmod R$$

となり，したがって $S \bmod R \sim T \bmod R$ が成り立つ．■

これらの補助定理を使って，次の定理が得られる．

定理 3.2.1 補助定理 3.2.3 の条件の下で，S と R から生成される多項式剰余列は，最初の要素 S を除いて T と R から生成される多項式剰余列に等しい．

3.2.2 Euclid 算法の数値的不安定性

2 つの多項式から構成される多項式剰余列が正規の場合，隣接する 2 つの剰余列の要素をそれぞれ F，G とすれば，$\deg F = \deg G + 1$ である．ある隣接する剰余列の要素 F，G において，F と G の主係数の絶対値が大きく異なるとき，多項式剰余列は不正規に近いという．この場合，桁溢れや桁落ちが起こり得る．桁溢れを防ぐ手法として，$F \bmod G$ と相似な部分終結式を枢軸選び付きの行列の上三角化によって求めることができる．Sasaki は，主係数消去による丸め誤差の成長を抑える対策として，規格化剰余多項式を提案している [29, 31]．しかし，いずれにしても，Euclid 算法の計算の過程で不正規に近い状態が発生した場合，桁落ちの原因となる．実例を挙げる前に，剰余多項式を求める行列算法を示す．

多項式 F，G をそれぞれ

$$\begin{aligned} F &= f_m z^m + f_{m-1} z^{m-1} + \cdots + f_0, \\ G &= g_{m-1} z^{m-1} + g_{m-2} z^{m-2} + \cdots + g_0 \end{aligned}$$

とするとき，$F \bmod G$ は部分終結式 $S_{m-2}(F,G)$ と相似である．このとき

$$\begin{bmatrix} F \\ zG \\ G \end{bmatrix} = \begin{bmatrix} f_m & f_{m-1} & f_{m-2}z^{m-2} + \cdots + f_0 \\ g_{m-1} & g_{m-2} & g_{m-3}z^{m-2} + \cdots + g_0 z \\ 0 & g_{m-1} & g_{m-2}z^{m-2} + \cdots + g_0 \end{bmatrix} \begin{bmatrix} z^m \\ z^{m-1} \\ 1 \end{bmatrix}$$

の右辺の係数行列の行列式が $S_{m-2}(F,G)$ である．

枢軸選び付きの Gauss の消去法によって上の行列を上三角化すれば，この右下隅の要素が $S_{m-2}(F,G)$ と相似となり，枢軸選びをしない場合に比べて精度は改善されるであろう．

Euclid 算法における除算（剰余の計算）を上の方法で実行する互除法を枢軸選び付き互除法ということにする．補助定理 3.2.2 により，この方法で得られる多項式列は，F と G から生成される多項式剰余列である．

しかしながら，多項式剰余列が不正規に近い場合，枢軸選びを行ってもさして効果はない．この事実を確かめるために，F と G のノルムは $O(1)$ として，G の主係数の絶対値が十分小さい $\varepsilon > 0$ であったとする．この場合，枢軸選びによる行の交換は行われない．行列式 $S_{m-2}(F,G)$ の上三角化による計算の過程を示せば

$$\begin{bmatrix} f_m & f_{m-1} & F \\ \varepsilon & g_{m-2} & zG \\ 0 & \varepsilon & G \end{bmatrix} \xrightarrow{(1)} \begin{bmatrix} f_m & f_{m-1} & F \\ 0 & g_{m-1}^{(0)} & G^{(0)} \\ 0 & \varepsilon & G \end{bmatrix} \xrightarrow{(2)} \begin{bmatrix} f_m & f_{m-1} & F \\ 0 & g_{m-1}^{(0)} & G^{(0)} \\ 0 & 0 & G^{(1)} \end{bmatrix}$$

(1)： 1 行 $\times(-\varepsilon/f_m)$ を 2 行に加える: $G^{(0)} = \mathrm{Elim}\{F, zG\}$

(2)： 2 行 $\times(-\varepsilon/g_{m-1}^{(0)})$ を 3 行に加える: $G^{(1)} = \mathrm{Elim}\{G^{(0)}, G\}$

となる．$G^{(0)}$ の主係数 $g_{m-1}^{(0)}$ の絶対値が ε より十分大と仮定する．このとき，消去の第二段階は枢軸交換なしで $G^{(1)}$ を求めることである．次に，$S_{m-3}(G, G^{(1)})$ と相似な多項式 $G \bmod G^{(1)}$ について考察してみる．

$G^{(0)}$，$G^{(1)}$ の定義より

$$\begin{aligned} G^{(0)} \bmod G^{(1)} &= \left\{ z\left(G^{(1)} + \frac{\varepsilon}{g_{m-1}^{(0)}} G^{(0)} \right) - \frac{\varepsilon}{f_m} F \right\} \bmod G^{(1)} \\ &= \varepsilon \cdot \left(\frac{z}{g_{m-1}^{(0)}} G^{(0)} - \frac{1}{f_m} F \right). \end{aligned}$$

したがって

$$\| G^{(0)} \bmod G^{(1)} \|_\infty = O(\varepsilon). \tag{3.16}$$

これを用いれば

$$\begin{aligned} \| G \bmod G^{(1)} \|_\infty &= \left\| \left(G^{(1)} + \frac{\varepsilon}{g_{m-1}^{(0)}} G^{(0)} \right) \bmod G^{(1)} \right\|_\infty \\ &= \left\| \varepsilon \cdot \frac{1}{g_{m-1}^{(0)}} \cdot G^{(0)} \bmod G^{(1)} \right\|_\infty \\ &= O(\varepsilon^2) \end{aligned} \tag{3.17}$$

となる．F，G のノルムは $O(1)$ であったので，$G^{(0)}$，$G^{(1)}$ のノルムも $O(1)$ となる．このとき，(3.16) 式において，$G^{(0)}$ を $G^{(1)}$ で割ったときの余りが $O(\varepsilon)$ であるので，この余りは必然的に桁落ちすることになる．上式 (3.17) においては，余りが $O(\varepsilon^2)$ となるので桁落ちは一層激しくなる．

この欠点は以下のようにして，いくらか改善することができる．補助定理 3.2.3 により，G と $G^{(1)}$ から生成される多項式剰余列と $G^{(0)}$ と $G^{(1)}$ から生成されるそれらは，互いに相似である．前述の議論により，後者の組合せが桁落ちの抑制に効果的であり，相対的に精度良く求まる．前者の組合せが Euclid 算法である．一般に，G と $G^{(0)}$ のうち主係数の絶対値の大きい方と $G^{(1)}$ との組合せが桁落ちを抑制する．この組合せに基づく算法を改良互除法と呼ぶことにする．多項式演算を用いて記述すれば次のようになる．

3.2.3 改良互除法

[改良互除法]

[初期設定] F, G $(\deg F > \deg G)$ を与える．

[手順 1] $d = \deg F - \deg G \ (\geq 1)$,
$G_0^{(0)} = z^{d-1} G$,
$G_1^{(0)} = \mathrm{Elim}\{F, zG_0^{(0)}\}$.

[手順 2]
$$\begin{bmatrix} i = 1, 2, \ldots . d-1 \\ G_0^{(i)} = z^{d-i-1} G, \\ G_1^{(i)} = \mathrm{Elim}\{G_0^{(i-1)}, G_1^{(i-1)}\}. \end{bmatrix}$$

[手順 3] $F \leftarrow \begin{cases} G_0^{(d-1)}, & |\mathrm{lc}(G_0^{(d-1)})| \geq |\mathrm{lc}(G_1^{(d-1)})| \\ G_1^{(d-1)}, & |\mathrm{lc}(G_0^{(d-1)})| < |\mathrm{lc}(G_1^{(d-1)})|, \end{cases}$
$G_1^{(d)} = \mathrm{Elim}\{G_0^{(d-1)}, G_1^{(d-1)}\}$,
$G \leftarrow G_1^{(d)}$.
$\deg G > 0$ ならば手順 1 へ.
$\deg G \leq 0$ ならば停止する. このとき
$\deg G = 0$ ならば与えられた F と G は互いに素.
$\deg G < 0$ ならば更新された F は, 与えられた F と G の最大公約因子.
更新される G の列は, 多項式剰余列である. このことは定理 3.2.1 より明らかである.

改良互除法の典型的な場合の計算量（代表的に乗除算回数）を評価する.
$\deg F = \deg G + 1 = m$ として, 多項式剰余列は正規とする. このときの計算量は約 m^2 回である.

数値例 3.2.1 互いに素な 2 つの多項式

$$F = z^6 + \frac{11}{12}(1+\varepsilon)z^5 + \frac{10}{11}z^4 + \frac{9}{10}z^3 + \frac{8}{9}z^2 + \frac{7}{8}z + \frac{6}{7},$$
$$G = z^5 + \frac{11}{12}z^4 + \frac{10}{11}(1+\varepsilon)z^3 + \frac{5}{6}z^2 + \frac{4}{5}z + \frac{3}{4}$$

に対して, Euclid 算法を適用する. $\varepsilon = 0$ のとき, 多項式剰余列は不正規となる. ε が十分小さいとき, 正規ではあるが不正規に近くなる. ここでは, $\varepsilon = 10^{-7}$ として, 不正規に近い状態を発生させた. $P_{-1} := F$, $P_0 := G$, $P_i := P_{i-2} \bmod P_{i-1}$ $(i \geq 1)$ で構成される多項式剰余列を 4 つの算法で倍精度計算した. Euclid 算法以外の方法で生成される多項式列は, 各 P_i と相似である. したがって, 多項式の係数が異なるときは, 両者を多項式のノルムで割って比較した. 表 3.1 は, Euclid 算法と規格化剰余を用いた互除法とを比較したものである. 下線部分は, 四倍精度と異なる部分である. いま, $|\mathrm{lc}(P_0)| = 1 \gg |\mathrm{lc}(P_1)|$ となっており, 不正規に近い状態であるので, P_1 と P_2 から P_3 を計算する段階で, 両方法ともに桁落ちが発生している. このときの多項式のノルムは両方法とも $O(10^{-12})$ である. 精度に関してはさほど変わらない. 表 3.2 は枢軸選び付き互除法と改良互除法とを比較したものである. ここでも, P_3 を計算する段階で, 両方法ともに桁落ちが発生している. 多項式のノルムを比較してみれば, 枢軸選び付き互除法では $O(10^{-12})$, 改良互除法では $O(10^{-6})$ となり, 後者の方は比較的桁落ちが抑制されている. 最終項 P_5 の精度は順番に 4 桁, 5 桁, 6 桁, 11 桁となっている. このことにより, 改良互除法は精度が改善されていることが分かる.

改良互除法の特徴を二点挙げる. 第一に, 多項式剰余列を部分終結式の枢軸選び付き上三角化により求める. 第二に, 主係数消去の際の桁落ちを防ぐために, 定理 3.2.1 を手順 3 に適用して, F と G を更新することにある. 次節で述べる算法は, 改良互除法の第一の点を一般化して, 多項式剰余列の部分列を部分終結式の枢軸選び付き上三角化により求める. それは, また改良互除法の第二の特徴をそのまま受け継ぐものである.

表 3.1: Euclid 算法と規格化剰余を用いた互除法との比較.

相似な剰余列要素	Euclid 算法	規格化剰余
P_1	$-1.7493686873168 \times 10^{-7}z^4$ $+6.6666583333325 \times 10^{-2}z^3$ $+8.8888812500000 \times 10^{-2}z^2$ $+0.12499992666667z$ $+0.85714278839286$	$-1.7493686873168 \times 10^{-7}z^4$ $+6.6666583333325 \times 10^{-2}z^3$ $+8.8888812500000 \times 10^{-2}z^2$ $+0.12499992666667z$ $+0.85714278839286$
P_2	$145229975132.52z^3$ $+193640079534.01z^2$ $+272310331004.51z$ $+1867238188723.8$	$6.6666816582431 \times 10^{-2}z^3$ $+8.8889140506440 \times 10^{-2}z^2$ $+0.12500217585257z$ $+0.85714278839320$
$P_2/\\|P_2\\|_\infty$	$7.7777958917912 \times 10^{-2}z^3$ $+1.0370400557540 \times 10^{-1}z^2$ $+0.14583588352519z + 1$	$7.7777958917912 \times 10^{-2}z^3$ $+1.0370400557540 \times 10^{-1}z^2$ $+0.14583588352519z + 1$
P_3	$4.90965601507\underline{30} \times 10^{-12}z^2$ $-3.17953996464\underline{84} \times 10^{-12}z$ $-2.76600964355\underline{11} \times 10^{-12}$	$4.90965601505\underline{91} \times 10^{-12}z^2$ $-3.17956772021\underline{50} \times 10^{-12}z$ $-2.76612066584\underline{58} \times 10^{-12}$
P_4	$540442552695.10z$ $+2029318777042.0$	$9.2230668404111 \times 10^{-12}z$ $+3.4631652195647 \times 10^{-11}$
$P_4/\\|P_4\\|_\infty$	$0.2663172286233\underline{2}z + 1$	$0.2663189959377\underline{8}z + 1$
P_5	$7.8396248252\underline{327} \times 10^{-11}$	$3.3451345163\underline{630} \times 10^{-11}$

表 3.2: 枢軸選び付き互除法と改良互除法との比較.

相似な剰余列要素	枢軸選び付き	改良互除法
P_1	$1.7493686873168 \times 10^{-7}z^4$ $-6.6666583333325 \times 10^{-2}z^3$ $-8.8888812500000 \times 10^{-2}z^2$ $-0.12499992666667z$ -0.85714278839286	$1.7493686873168 \times 10^{-7}z^4$ $-6.6666583333325 \times 10^{-2}z^3$ $-8.8888812500000 \times 10^{-2}z^2$ $-0.12499992666667z$ -0.85714278839286
P_2	$-6.6666816582431 \times 10^{-2}z^3$ $-8.8889140506440 \times 10^{-2}z^2$ $-0.12500217585257z$ -0.85714278839320	$-6.6666816582431 \times 10^{-2}z^3$ $-8.8889140506440 \times 10^{-2}z^2$ $-0.12500217585257z$ -0.85714278839320
P_3	$4.90965601507\underline{30} \times 10^{-12}z^2$ $-3.17956772022\underline{40} \times 10^{-12}z$ $-2.76600964355\underline{11} \times 10^{-12}$	$1.87101935724\underline{67} \times 10^{-6}z^2$ $-1.21170844989\underline{34} \times 10^{-6}z$ $-1.05416041998\underline{83} \times 10^{-6}$
$P_3/\|P_3\|_\infty$	$z^2 - 0.6476151710\underline{9600}z$ $-0.5633815556\underline{6485}$	$z^2 - 0.6476194087\underline{4650}z$ $-0.5634150260\underline{9549}$
P_4	$-9.2230107954\underline{371} \times 10^{-12}z$ $-3.4631541173\underline{442} \times 10^{-11}$	$-3.5148290684\underline{207} \times 10^{-6}z$ $-1.3197760419\underline{092} \times 10^{-5}$
$P_4/\|P_4\|_\infty$	$-0.2663182313\underline{8469}z - 1$	$-0.2663201146\underline{8674}z - 1$
P_5	$-3.3451291576\underline{621} \times 10^{-11}$	$-2.2236235038\underline{730} \times 10^{-5}$

3.3 算法

多項式剰余列の生成は部分終結式の反復計算と解釈できる．部分終結式を枢軸選び付きの Gauss の消去法によって求める．

我々は，2 つの多項式

$$F = f_m z^m + f_{m-1}z^{m-1} + \cdots + f_0 \tag{3.18}$$
$$G = g_n z^n + g_{n-1}z^{n-1} + \cdots + g_0,\ f_m \neq 0,\ g_n \neq 0,\ n \leq m \tag{3.19}$$

に対して，次の手順で計算を行う．

[算法]

0) $\deg F > \deg G$ ならば 1) へ．
$\deg F = \deg G$ の場合
$\mathrm{Piv}\{F, G\}$, $G \leftarrow \mathrm{Elim}\{F, G\}$ によって F と G を更新して 1) へ．

1) $\deg G \leq 0$ ならば停止する．このとき
$\deg G < 0 \Longrightarrow F$ は与えられた F と G の最大公約因子．
$\deg G = 0 \Longrightarrow$ 与えられた F と G は互いに素．

2) $\deg G \geq 1$ の場合
$d := \deg F - \deg G\ (\geq 1)$ とおく．

[手順 1] $G_0^{(0)} = z^{d-1}G$,
$\left[\begin{array}{l} j = 1, 2, \ldots, l+1 \\ G_j^{(0)} = \mathrm{Elim}\{F, zG_{j-1}^{(0)}\}. \end{array}\right.$
注. l は別途定める．

[手順 2] $\left[\begin{array}{l} i = 1, 2, \ldots, d-1 \\ \mathrm{Piv}\{G_0^{(i-1)}, G_1^{(i-1)}, \ldots, G_l^{(i-1)}\}, \\ G_0^{(i)} = z^{d-i-1}G, \\ \left[\begin{array}{l} j = 1, 2, \ldots, l+1 \\ G_j^{(i)} = \mathrm{Elim}\{G_0^{(i-1)}, G_j^{(i-1)}\}. \end{array}\right. \end{array}\right.$

[手順 3] $\left[\begin{array}{l} k = 0, 1, \ldots, l-1 \\ \mathrm{Piv}\{G_k^{(d+k-1)}, G_{k+1}^{(d+k-1)}, \ldots, G_l^{(d+k-1)}\}, \\ \left[\begin{array}{l} j = k+1, k+2, \ldots, l+1 \\ G_j^{(d+k)} = \mathrm{Elim}\{G_k^{(d+k-1)}, G_j^{(d+k-1)}\}. \end{array}\right. \end{array}\right.$

[手順 4] $F \leftarrow \begin{cases} G_l^{(d+l-1)}, & |\mathrm{lc}(G_l^{(d+l-1)})| \geq |\mathrm{lc}(G_{l+1}^{(d+l-1)})| \\ G_{l+1}^{(d+l-1)}, & |\mathrm{lc}(G_l^{(d+l-1)})| < |\mathrm{lc}(G_{l+1}^{(d+l-1)})|, \end{cases}$
$G_{l+1}^{(d+l)} = \mathrm{Elim}\{G_l^{(d+l-1)}, G_{l+1}^{(d+l-1)}\}$,
$G \leftarrow G_{l+1}^{(d+l)}$.
1) へ戻る．

手順 3 において，枢軸多項式の次数が 1 次ずつ減少することを仮定する．このことは枢軸多項式の主係数を調べれば確かめることができる．

l と l の上限 L の選び方により方法は分かれる．常に $0 \leq l \leq L \leq \deg G - 1$ である．

l の選択法 1 : 一定の l を与える．例えば $l = 0$，または 1．特に，$l = 0$ のとき改良互除法が得られる．

l の選択法 2 : $|\mathrm{lc}(G_l^{(0)})| \geq |\mathrm{lc}(F)|$ を満たす最小の整数．
そのような整数が存在しなければ $l = L$ とする．
l をこのようにとる理由は不正規に近い状態を避けるためである．
以後，我々は，$L = \deg G - 1$ とする．

l の選択法 2 が安定性の点で l の選択法 1 より優れている．

本方法で得られた多項式列を二重配列に並べれば次のようになる．

$$
\begin{array}{cccccc}
F & & & & & \\
G_0^{(0)} & G_1^{(0)} & \cdots & \cdots & G_l^{(0)} & G_{l+1}^{(0)} \\
G_0^{(1)} & G_1^{(1)} & \cdots & \cdots & G_l^{(1)} & G_{l+1}^{(1)} \\
\vdots & \vdots & \cdots & \cdots & \vdots & \vdots \\
G_0^{(d-1)} & G_1^{(d-1)} & \cdots & \cdots & G_l^{(d-1)} & G_{l+1}^{(d-1)} \\
 & G_1^{(d)} & \cdots & \cdots & G_l^{(d)} & G_{l+1}^{(d)} \\
 & & \ddots & & \vdots & \vdots \\
 & & & & G_l^{(d+l-1)} & G_{l+1}^{(d+l-1)} \\
 & & & & & G_{l+1}^{(d+l)}
\end{array}
$$

本算法により生成される多項式列は，次の意味を持つ．

定理 3.3.1 本算法で得られた $G_l^{(d+l-1)}$，$G_{l+1}^{(d+l)}$ は，(3.18)，(3.19) で与えられる多項式 F，G の部分終結式 $S_{n-l}(F,G)$，$S_{n-l-1}(F,G)$ とそれぞれ相似である．

[証明]　本算法は，部分終結式の計算に他ならないことを示す．本算法から生成される多項式 $G_j^{(i)}$ を

$$G_j^{(i)} := \sum_{k=0}^{m-i-1} g_{j,k}^{(i)} z^k$$

とおく．

手順 1 は，上三角化する部分終結式の次数を定め，部分終結式を構成する F 以外の多項式列の次数を全て $m-1$ 次以下にする過程である．

$$
S_{n-l-1}(F,G) \sim \begin{vmatrix}
f_m & f_{m-1} & \cdots & \cdots & \cdots & \cdots & F \\
 & g_{l+1,m-1}^{(0)} & \cdots & \cdots & \cdots & \cdots & G_{l+1}^{(0)} \\
 & g_{l,m-1}^{(0)} & \cdots & \cdots & \cdots & \cdots & G_l^{(0)} \\
 & \vdots & & & & & \vdots \\
 & g_{1,m-1}^{(0)} & \cdots & \cdots & \cdots & \cdots & G_1^{(0)} \\
 & g_{0,m-1}^{(0)} & \cdots & \cdots & \cdots & \cdots & G_0^{(0)} \\
 & & g_n & \cdots & \cdots & \cdots & Gz^{d-2} \\
 & & & \ddots & & & \vdots \\
 0 & & & & g_n & \cdots & G
\end{vmatrix} . \qquad (3.20)
$$

ここで，上式右辺の行列式は，部分終結式 $S_{n-l-1}(F,G)$ の行列成分に手順 1 による掃き出し後，第 1 行から第 l 行までを省略した表現になっている．上式の行列成分の 2 行目と右端 2 列目の成分を除いて得られる小行列式は，部分終結式 $S_{n-l}(F,G)$ と相似である．

手順 2 は，上式の枢軸選び付き上三角化の途中経過であり，$G_0^{(d-1)}\ (=G)$ と同じ次数の多項式列 $G_1^{(d-1)}$，$G_2^{(d-1)}$，…，$G_{l+1}^{(d-1)}$ を計算する過程である．ただし，下つき添字が $l+1$ である多項式を除いた残りの多項式列で枢軸選びを行っていることに注意する．選ばれた枢軸多項式と下つき添字が $l+1$ の多項式で主係数消去するときは，演算 Elim を使っているので，用いられる乗数の絶対値は 1 以下となる．1 つの多項式だけ残して枢軸選びを行う理由は，部分終結式 $S_{n-l-1}(F,G)$ と同時に $S_{n-l}(F,G)$ を計算するためである．$S_{n-l}(F,G)$ に関しては，行を入れ換える枢軸選びを完全に行っている．

$$
(3.20)\text{式} \sim \begin{vmatrix}
f_m & f_{m-1} & \cdots & \cdots & \cdots & \cdots & \cdots & F \\
 & g_{l+1,m-1}^{(0)} & \cdots & \cdots & \cdots & \cdots & \cdots & G_{l+1}^{(0)} \\
 & & g_{0,m-2}^{(1)} & \cdots & \cdots & \cdots & \cdots & G_0^{(1)} \\
 & & & \ddots & & & & \vdots \\
 & & & & g_{0,n+1}^{(d-2)} & g_{0,n}^{(d-2)} & \cdots & G_0^{(d-2)} \\
 & & & & & g_{0,n}^{(d-1)} & \cdots & G_0^{(d-1)} \\
 & & & & & g_{1,n}^{(d-1)} & \cdots & G_1^{(d-1)} \\
 & & & & & \vdots & \cdots & \vdots \\
 0 & & & & & g_{l+1,n}^{(d-1)} & \cdots & G_{l+1}^{(d-1)}
\end{vmatrix} . \qquad (3.21)
$$

上式の 3 行目以下の右端の列成分 $G_0^{(1)}$，$G_0^{(2)}$，…，$G_0^{(d-2)}$ は，主係数消去の際の枢軸多項式である．

手順 3 は，上式の右下隅 $l+2$ 次の行列を枢軸選びをしながら右下隅 2 次の行列まで上三角化する過程である．

$$
(3.21)\text{式} \sim \begin{vmatrix}
g_{0,n}^{(d-1)} & \cdots & \cdots & \cdots & \cdots & G_0^{(d-1)} \\
 & g_{1,n-1}^{(d)} & \cdots & \cdots & \cdots & G_1^{(d)} \\
 & & \ddots & & & \vdots \\
 & & & g_{l-1,n-l+1}^{(d+l-2)} & g_{l-1,n-l}^{(d+l-2)} & G_{l-1}^{(d+l-2)} \\
 & & & & g_{l,n-l}^{(d+l-1)} & G_l^{(d+l-1)} \\
 0 & & & & g_{l+1,n-l}^{(d+l-1)} & G_{l+1}^{(d+l-1)}
\end{vmatrix} . \qquad (3.22)
$$

上式の右端から 2 列目と下端の行を除いた行列式は，$S_{n-l}(F,G)$ と相似である．したがって，この段階で，$S_{n-l}(F,G)$ と相似な多項式 $G_l^{(d+l-1)}$ が得られたことになる．

手順 4 は，上式の右下隅の 2 次の行列の上三角化である．

$$
(3.22)\text{式} \sim \begin{vmatrix}
g_{0,n}^{(d-1)} & \cdots & \cdots & \cdots & \cdots & G_0^{(d-1)} \\
 & g_{1,n-1}^{(d)} & \cdots & \cdots & \cdots & G_1^{(d)} \\
 & & \ddots & & & \vdots \\
 & & & g_{l-1,n-l+1}^{(d+l-2)} & g_{l-1,n-l}^{(d+l-2)} & G_{l-1}^{(d+l-2)} \\
 & & & & g_{l,n-l}^{(d+l-1)} & G_l^{(d+l-1)} \\
 0 & & & & & G_{l+1}^{(d+l)}
\end{vmatrix} . \qquad (3.23)
$$

明らかに，右下隅の要素 $G_{l+1}^{(d+l)}$ は，$S_{n-l-1}(F,G)$ と相似である．よって，隣接する剰余列の要素 F，G より，$S_{n-l}(F,G)$，$S_{n-l-1}(F,G)$ が得られる．■

本算法では，従来の Euclid 算法に対し，主に次の 3 つの改良を行い，数値的安定性を向上させた．

1. 主係数消去の演算 Elim を使うことによって，現れる乗数の絶対値を 1 以下にした．
2. 不正規に近い場合を避けるために，パラメータ l を適切に選んだ．
3. 定理 3.2.1 を手順 4 に適用して，桁落ちを避けるようにした．

本方法で得られる多項式列は，多項式剰余列の中で数値的に安定に求めやすい部分列である．

• 零判定

多項式剰余列の生成において，多項式およびその主係数の零判定は一般に難しい．前述の算法の手順 4 における多項式および主係数の零判定を次のように行った．

倍精度演算の下で零判定するために必要な定数 ε および γ を

$$\varepsilon := 10^{-12},\ \gamma =:= \max(\|F\|_\infty, \|G\|_\infty)$$

とする．簡単のために，手順 4 の 2 つの多項式 $G_l^{(d+l-1)}$，$G_{l+1}^{(d+l-1)}$ の主係数消去において，枢軸多項式を P，他方の多項式を Q とし，消去の結果 $G_{l+1}^{(d+l)}$ を R とおけば

$$R = Q - \frac{\mathrm{lc}(Q)}{\mathrm{lc}(P)}P$$

と書ける．R，$\mathrm{lc}(R)$ が次の条件を満たすとき，それぞれを零とみなす．

- 多項式 R の零判定

$$\frac{|\mathrm{lc}(P)|}{\gamma} \times \frac{\|R\|_\infty}{\gamma} \le \varepsilon.$$

- 多項式 R の主係数 $\mathrm{lc}(R)$ の零判定

$$\frac{|\mathrm{lc}(P)|}{\gamma} \times \frac{|\mathrm{lc}(R)|}{\gamma} \le \varepsilon.$$

• 数値例

3.2 節の数値例 3.2.1 と同じ例に対して，本算法（l の選択法 2 ）と改良互除法とを比較した結果を 表 3.3 に示す．本方法では，隣接する多項式 P_0，P_1 から次の隣接する多項式を安定に求めるために，パラメータ l は自動的に 3 に選択された．したがって，P_2，P_3 は計算されずに P_4，P_5 を求めている．先の 4 つの算法で起こったような桁落ちはなく，改良互除法に比べて精度も良好である．

表 3.3: 改良互除法と本算法との比較．

相似な剰余列要素	改良互除法	本算法
P_1	$1.7493686873168 \times 10^{-7}z^4$ $-6.6666583333325 \times 10^{-2}z^3$ $-8.8888812500000 \times 10^{-2}z^2$ $-0.12499992666667z$ -0.85714278839286	$1.7493686873168 \times 10^{-7}z^4$ $-6.6666583333325 \times 10^{-2}z^3$ $-8.8888812500000 \times 10^{-2}z^2$ $-0.12499992666667z$ -0.85714278839286
P_2	$-6.6666816582431 \times 10^{-2}z^3$ $-8.8889140506440 \times 10^{-2}z^2$ $-0.12500217585257z$ -0.85714278839320	
P_3	$1.8710193572467 \times 10^{-6}z^2$ $-1.2117084498934 \times 10^{-6}z$ $-1.0541604199883 \times 10^{-6}$	
P_4	$-3.5148290684207 \times 10^{-6}z$ $-1.3197760419092 \times 10^{-5}$	$-0.23056649210443z$ -0.86574944736120
$P_4/\|P_4\|_\infty$	$-0.26632011468674z - 1$	$-0.26632011467891z - 1$
P_5	$-2.2236235038730 \times 10^{-5}$	-0.86223029085074

3.4 拡張算法

前節において我々は，多項式剰余列の不正規に近い状態を避けて，数値的に安定に求まる剰余列の要素だけを計算する算法，すなわち多項式剰余列の安定な生成法を提案した．

本節では，2 つの多項式 F，G ($\deg F \ge \deg G$) の多項式剰余列の要素 P_i と同時に次の条件

$$\left.\begin{aligned} P_i &= A_iF + B_iG, \\ \deg P_i &\le \deg G - \deg A_i - 1 \end{aligned}\right\} \tag{3.24}$$

を満たす多項式 A_i，B_i を求める安定な拡張算法を提案する．

まず，本章の最初に導入した演算子 Piv と Elim を次のように拡張定義する．

$\{G_i\}_{i=0}^k$ は同じ次数の多項式列とする．各 G_i に対して，2 つの多項式 A_i，B_i が付随的に定まるものとする．そこで多項式の組の列 $\left\{\begin{bmatrix} G_i \\ A_i \\ B_i \end{bmatrix}\right\}_{i=0}^k$ に対して次の演算を定義する．

Piv　$\{G_i\}_{i=0}^k$ の中で主係数の絶対値最大の多項式を G_{i_0} としたときに，i_0 番目の組

$\begin{bmatrix} G_{i_0} \\ A_{i_0} \\ B_{i_0} \end{bmatrix}$ と 0 番目の組 $\begin{bmatrix} G_0 \\ A_0 \\ B_0 \end{bmatrix}$ を交換する操作.

Elim 同じ次数の多項式 G_0, G_1 の組 $\begin{bmatrix} G_0 \\ A_0 \\ B_0 \end{bmatrix}$, $\begin{bmatrix} G_1 \\ A_1 \\ B_1 \end{bmatrix}$ に対して，次で定義される主係数消去の演算.

$$\mathrm{Elim}\left\{\begin{bmatrix} G_0 \\ A_0 \\ B_0 \end{bmatrix}, \begin{bmatrix} G_1 \\ A_1 \\ B_1 \end{bmatrix}\right\} := \begin{cases} \begin{bmatrix} G_1 \\ A_1 \\ B_1 \end{bmatrix} - \dfrac{\mathrm{lc}(G_1)}{\mathrm{lc}(G_0)} \begin{bmatrix} G_0 \\ A_0 \\ B_0 \end{bmatrix}, & |\mathrm{lc}(G_0)| > \mathrm{lc}(G_1)| \\ \begin{bmatrix} G_0 \\ A_0 \\ B_0 \end{bmatrix} - \dfrac{\mathrm{lc}(G_0)}{\mathrm{lc}(G_1)} \begin{bmatrix} G_1 \\ A_1 \\ B_1 \end{bmatrix}, & |\mathrm{lc}(G_0)| \le |\mathrm{lc}(G_1)|. \end{cases}$$

3.4.1 拡張改良互除法

改良互除法の拡張算法は，3.2 節の改良互除法に基づき多項式剰余列 P_i およびそれに付随する多項式 A_i, B_i を同時に求める算法である．以後，この算法を拡張改良互除法とよぶことにする．

[拡張改良互除法]

[入力] 2 つの多項式 F, G, $\deg F > \deg G$.

[出力] 多項式剰余列 $\{P_i\}_{i=-1}^{t}$ および (3.24) を満たす多項式 A_i, B_i.
ここで，多項式 P_i, A_i, B_i はそれぞれ以下の算法によって更新される多項式 G, A_G, B_G のことである.

[初期設定] $\begin{bmatrix} F \\ A_F \\ B_F \end{bmatrix} \leftarrow \begin{bmatrix} F \\ 1 \\ 0 \end{bmatrix}$, $\begin{bmatrix} G \\ A_G \\ B_G \end{bmatrix} \leftarrow \begin{bmatrix} G \\ 0 \\ 1 \end{bmatrix}$.

[手順 1] $d = \deg F - \deg G\ (\ge 1)$,

$$\begin{bmatrix} G_0^{(0)} \\ A_0^{(0)} \\ B_0^{(0)} \end{bmatrix} = z^{d-1} \begin{bmatrix} G \\ A_G \\ B_G \end{bmatrix},$$

$$\begin{bmatrix} G_1^{(0)} \\ A_1^{(0)} \\ B_1^{(0)} \end{bmatrix} = \mathrm{Elim}\left\{\begin{bmatrix} F \\ A_F \\ B_F \end{bmatrix}, z\begin{bmatrix} G_0^{(0)} \\ A_0^{(0)} \\ B_0^{(0)} \end{bmatrix}\right\}.$$

[手順 2]
$$\begin{cases} i = 1, 2, \ldots . d-1 \\ \begin{bmatrix} G_0^{(i)} \\ A_0^{(i)} \\ B_0^{(i)} \end{bmatrix} = z^{d-i-1} \begin{bmatrix} G \\ A_G \\ B_G \end{bmatrix}, \\ \begin{bmatrix} G_1^{(i)} \\ A_1^{(i)} \\ B_1^{(i)} \end{bmatrix} = \mathrm{Elim}\left\{\begin{bmatrix} G_0^{(i-1)} \\ A_0^{(i-1)} \\ B_0^{(i-1)} \end{bmatrix}, \begin{bmatrix} G_1^{(i-1)} \\ A_0^{(i-1)} \\ B_0^{(i-1)} \end{bmatrix}\right\}. \end{cases}$$

[手順 3]
$$\begin{bmatrix} G_1^{(d)} \\ A_1^{(d)} \\ B_1^{(d)} \end{bmatrix} = \mathrm{Elim}\left\{\begin{bmatrix} G_0^{(d-1)} \\ A_0^{(d-1)} \\ B_0^{(d-1)} \end{bmatrix}, \begin{bmatrix} G_1^{(d-1)} \\ A_1^{(d-1)} \\ B_1^{(d-1)} \end{bmatrix}\right\},$$

$$\begin{bmatrix} G \\ A_G \\ B_G \end{bmatrix} \leftarrow \begin{bmatrix} G_1^{(d)} \\ A_1^{(d)} \\ B_1^{(d)} \end{bmatrix},$$

$$\begin{bmatrix} F \\ A_F \\ B_F \end{bmatrix} \leftarrow \begin{cases} \begin{bmatrix} G_0^{(d-1)} \\ A_0^{(d-1)} \\ B_0^{(d-1)} \end{bmatrix}, & |\mathrm{lc}(G_0^{(d-1)})| \ge |\mathrm{lc}(G_1^{(d-1)})| \\ & \text{または}, \deg G_1^{(d)} < 0 \\ \begin{bmatrix} G_1^{(d-1)} \\ A_0^{(d-1)} \\ B_0^{(d-1)} \end{bmatrix}, & |\mathrm{lc}(G_0^{(d-1)})| < |\mathrm{lc}(G_1^{(d-1)})|. \end{cases}$$

$\deg G > 0$ ならば手順 1 へ.
$\deg G \le 0$ ならば停止する．このとき
$\deg G = 0$ ならば与えられた F と G は互いに素.
$\deg G < 0$ ならば与えられた F と G の最大公約因子は F であって，F とそれに付随する多項式 A_F, B_F は，関係式 (3.24) を満足する.
手順 3 において更新される G の列は，多項式剰余列であり，それに付随する多項式 A_G, B_G は (3.24) を満足する.

数値例 3.2.1 で取り挙げた多項式剰余列が不正規に近い問題に対して，拡張 Euclid 算法と拡張改良互除法の計算結果を示す．互いに素な多項式

$$\begin{aligned} F &= z^6 + \frac{11}{12}(1+\varepsilon)z^5 + \frac{10}{11}z^4 + \frac{9}{10}z^3 + \frac{8}{9}z^2 + \frac{7}{8}z + \frac{6}{7}, \\ G &= z^5 + \frac{11}{12}z^4 + \frac{10}{11}(1+\varepsilon)z^3 + \frac{5}{6}z^2 + \frac{4}{5}z + \frac{3}{4}, \quad \varepsilon = 10^{-7} \end{aligned}$$

について，(3.24) を満たす多項式列を倍精度演算で計算した．下線部分は，四倍精度演算で計算した結果と比較して異なる部分である.

拡張 Euclid 算法の計算結果

$$
\begin{aligned}
P_1 &= 0.85714278839286 + 0.12499992666667z + 8.8888812500000 \times 10^{-2}z^2 \\
&\quad + 6.6666583333325 \times 10^{-2}z^3 - 1.7493686873168 \times 10^{-7}z^4, \\
A_1 &= 1.0000000000000, \\
B_1 &= -9.1666666701684 \times 10^{-8} - 1.0000000000000z, \\
P_2 &= 1867238188723.8 + 272310331004.51z + 193640079534.01z^2 \\
&\quad + 145229975132.52z^3, \\
A_2 &= 2178444728239.7 + 5716347.8873847z, \\
B_2 &= -199689.766831599 - 2178444728240.2z - 5716347.8873847z^2, \\
P_3 &= -2.7660096435511 \times 10^{-12} - 3.1795399646484 \times 10^{-12}z \\
&\quad + 4.9096560150730 \times 10^{-12}z^2, \\
A_3 &= -2.8252955530661 \times 10^{-12} - 2.8690182049926 \times 10^{-12}z \\
&\quad + 6.8856308698500 \times 10^{-12}z^2, \\
B_3 &= -4.5904283126396 \times 10^{-13} + 2.8255175976710 \times 10^{-12}z \\
&\quad + 2.8690175739531 \times 10^{-12}z^2 - 6.8856308698500 \times 10^{-12}z^3, \\
P_4 &= 2029318777042.0 + 540442552695.10z, \\
A_4 &= 2343999308456.5 + 251695919102.67z - 318612175627.89z^2 \\
&\quad - 203680257217.60z^3, \\
B_4 &= 26898452788.896 - 2330433612974.3z - 251702450311.18z^2 \\
&\quad + 318612194294.34z^3 + 203680257217.60z^4, \\
P_5 &= 7.8396248252327 \times 10^{-11}, \\
A_5 &= 9.0922556451738 \times 10^{-11} - 1.4096569109771 \times 10^{-11}z \\
&\quad - 8.1437428840391 \times 10^{-12}z^2 - 5.2517214628999 \times 10^{-12}z^3 \\
&\quad + 1.8503354242059 \times 10^{-12}z^4, \\
B_5 &= 6.1675619398069 \times 10^{-13} - 9.0624136008563 \times 10^{-11}z \\
&\quad + 1.3973069561856 \times 10^{-11}z^2 + 8.1438029634327 \times 10^{-12}z^3 \\
&\quad + 5.2517212933243 \times 10^{-12}z^4 - 1.8503354242059 \times 10^{-12}z^5.
\end{aligned}
$$

拡張改良互除法の計算結果

$$
\begin{aligned}
P_1 &\sim -0.85714278839286 - 0.12499992666667z - 8.8888812500000 \times 10^{-2}z^2 \\
&\quad - 6.6666583333325 \times 10^{-2}z^3 + 1.7493686873168 \times 10^{-7}z^4,
\end{aligned}
$$

$$
\begin{aligned}
A_1 &\sim -1.0000000000000, \\
B_1 &\sim 9.1666666701684 \times 10^{-8} + 1.0000000000000z, \\
P_2 &\sim -0.85714278839320 - 0.12500217585257z - 8.8889140506440 \times 10^{-2}z^2 \\
&\quad - 6.6666816582431 \times 10^{-2}z^3, \\
A_2 &\sim -1.0000000000000 - 2.6240499991955 \times 10^{-6}z, \\
B_2 &\sim 9.1666207658594 \times 10^{-8} + 1.0000000000002z + 2.6240499991955 \times 10^{-6}z^2, \\
P_3 &\sim -1.0541604199883 \times 10^{-6} - 1.2117084498934 \times 10^{-6}z \\
&\quad + 1.8710193572467 \times 10^{-6}z^2, \\
A_3 &\sim -1.0767841495461 \times 10^{-6} - 1.0933550069937 \times 10^{-6}z \\
&\quad + 2.6240471301847 \times 10^{-6}z^2, \\
B_3 &\sim -1.7493677002696 \times 10^{-7} + 1.0767847088129 \times 10^{-6}z \\
&\quad + 1.0933547665194 \times 10^{-6}z^2 - 2.6240471301847 \times 10^{-6}z^3, \\
P_4 &\sim -1.3197760419092 \times 10^{-5} - 3.5148290684207 \times 10^{-6}z, \\
A_4 &\sim -1.5244318618183 \times 10^{-5} - 1.6369607929481 \times 10^{-6}z \\
&\quad + 2.0721134290430 \times 10^{-6}z^2 + 1.3246384159480 \times 10^{-6}z^3, \\
B_4 &\sim -1.7493547134280 \times 10^{-7} + 1.5156009800966 \times 10^{-5}z \\
&\quad + 1.6369608347958 \times 10^{-6}z^2 - 2.0721135504362 \times 10^{-6}z^3 \\
&\quad - 1.3246384159480 \times 10^{-6}z^4, \\
P_5 &\sim -2.2236235038730 \times 10^{-5}, \\
A_5 &\sim -2.5789206520211^{-5} + 3.9983413963 18 \times 10^{-6}z \\
&\quad + 2.3098842390963 \times 10^{-6}z^2 + 1.4895809204313 \times 10^{-6}z^3 \\
&\quad - 5.2483054930924 \times 10^{-7}z^4, \\
B_5 &\sim -1.7493450473291 \times 10^{-7} + 2.5704479681037 \times 10^{-5}z \\
&\quad - 3.9633453470126 \times 10^{-6}z^2 - 2.3098844674227 \times 10^{-6}z^3 \\
&\quad - 1.4895808723345 \times 10^{-6}z^4 + 5.2483054930924 \times 10^{-7}z^5.
\end{aligned}
$$

以上の数値例から，最終項の精度を比較してみれば，従来の拡張 Euclid 算法では約 4 桁，これに対して，拡張改良互除法では約 9 桁である．この結果から，拡張改良互除法は拡張 Euclid 算法より改良されているといえる．

3.4.2 拡張算法（部分終結式算法）

3.3 節で述べた多項式剰余列の安定な生成法は，拡張改良互除法と同じ意味において拡張される．

多項式 F，G は与えられる．$\deg F \geq \deg G$ とする．

[安定な拡張算法]

[初期設定] $\begin{bmatrix} F \\ A_F \\ B_F \end{bmatrix} \leftarrow \begin{bmatrix} F \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} G \\ A_G \\ B_G \end{bmatrix} \leftarrow \begin{bmatrix} G \\ 0 \\ 1 \end{bmatrix}.$

0) $\deg F > \deg G$ ならば 1) へ.
$\deg F = \deg G$ の場合

$$\mathrm{Piv}\left\{\begin{bmatrix} F \\ A_F \\ B_F \end{bmatrix}, \begin{bmatrix} G \\ A_G \\ B_G \end{bmatrix}\right\}, \begin{bmatrix} G \\ A_G \\ B_G \end{bmatrix} \leftarrow \mathrm{Elim}\left\{\begin{bmatrix} F \\ A_F \\ B_F \end{bmatrix}, \begin{bmatrix} G \\ A_G \\ B_G \end{bmatrix}\right\}$$

によって $\begin{bmatrix} F \\ A_F \\ B_F \end{bmatrix}$ と $\begin{bmatrix} G \\ A_G \\ B_G \end{bmatrix}$ を更新して 1) へ.

1) $\deg G \leq 0$ ならば停止する. このとき
$\deg G < 0$ ならば与えられた F と G の最大公約因子は F であって, F とそれに付随する多項式 A_F, B_F は (3.24) を満足する.
$\deg G = 0$ ならば与えられた F と G は互いに素.

2) $\deg G \geq 1$ の場合
$d := \deg F - \deg G\ (\geq 1)$ とおく.

[手順 1] $\begin{bmatrix} G_0^{(0)} \\ A_0^{(0)} \\ B_0^{(0)} \end{bmatrix} = z^{d-1} \begin{bmatrix} G \\ A_G \\ B_G \end{bmatrix},$

$$\left[\begin{array}{l} j = 1, 2, \ldots, l+1 \\ \left[\begin{array}{c} G_j^{(0)} \\ A_j^{(0)} \\ B_j^{(0)} \end{array}\right] = \mathrm{Elim}\left\{\begin{bmatrix} F \\ A_F \\ B_F \end{bmatrix}, z\begin{bmatrix} G_{j-1}^{(0)} \\ A_{j-1}^{(0)} \\ B_{j-1}^{(0)} \end{bmatrix}\right\}. \end{array}\right.$$

注. l の選択法については 3.3 節を参照のこと.

[手順 2]
$$\left[\begin{array}{l} i = 1, 2, \ldots, d-1 \\ \mathrm{Piv}\left\{\begin{bmatrix} G_0^{(i-1)} \\ A_0^{(i-1)} \\ B_0^{(i-1)} \end{bmatrix}, \begin{bmatrix} G_1^{(i-1)} \\ A_1^{(i-1)} \\ B_1^{(i-1)} \end{bmatrix}, \ldots, \begin{bmatrix} G_l^{(i-1)} \\ A_l^{(i-1)} \\ B_l^{(i-1)} \end{bmatrix}\right\}, \\ \begin{bmatrix} G_0^{(i)} \\ A_0^{(i)} \\ B_0^{(i)} \end{bmatrix} = z^{d-i-1} \begin{bmatrix} G \\ A_G \\ B_G \end{bmatrix}, \\ \left[\begin{array}{l} j = 1, 2, \ldots, l+1 \\ \left[\begin{array}{c} G_j^{(i)} \\ A_j^{(i)} \\ B_j^{(i)} \end{array}\right] = \mathrm{Elim}\left\{\begin{bmatrix} G_0^{(i-1)} \\ A_0^{(i-1)} \\ B_0^{(i-1)} \end{bmatrix}, \begin{bmatrix} G_j^{(i-1)} \\ A_j^{(i-1)} \\ B_j^{(i-1)} \end{bmatrix}\right\}. \end{array}\right. \end{array}\right.$$

[手順 3]
$$\left[\begin{array}{l} k = 0, 1, \ldots, l-1 \\ \mathrm{Piv}\left\{\begin{bmatrix} G_k^{(d+k-1)} \\ A_k^{(d+k-1)} \\ B_k^{(d+k-1)} \end{bmatrix}, \begin{bmatrix} G_{k+1}^{(d+k-1)} \\ A_{k+1}^{(d+k-1)} \\ B_{k+1}^{(d+k-1)} \end{bmatrix}, \ldots, \begin{bmatrix} G_l^{(d+k-1)} \\ A_l^{(d+k-1)} \\ B_l^{(d+k-1)} \end{bmatrix}\right\}, \\ \left[\begin{array}{l} j = k+1, k+2, \ldots, l+1 \\ \left[\begin{array}{c} G_j^{(d+k)} \\ A_j^{(d+k)} \\ B_j^{(d+k)} \end{array}\right] = \mathrm{Elim}\left\{\begin{bmatrix} G_k^{(d+k-1)} \\ A_k^{(d+k-1)} \\ B_k^{(d+k-1)} \end{bmatrix}, \begin{bmatrix} G_j^{(d+k-1)} \\ A_j^{(d+k-1)} \\ B_j^{(d+k-1)} \end{bmatrix}\right\}. \end{array}\right. \end{array}\right.$$

[手順 4] $\begin{bmatrix} G_{l+1}^{(d+l)} \\ A_{l+1}^{(d+l)} \\ B_{l+1}^{(d+l)} \end{bmatrix} = \mathrm{Elim}\left\{\begin{bmatrix} G_l^{(d+l-1)} \\ A_l^{(d+l-1)} \\ B_l^{(d+l-1)} \end{bmatrix}, \begin{bmatrix} G_{l+1}^{(d+l-1)} \\ A_{l+1}^{(d+l-1)} \\ B_{l+1}^{(d+l-1)} \end{bmatrix}\right\},$

$$\begin{bmatrix} G \\ A_G \\ B_G \end{bmatrix} \leftarrow \begin{bmatrix} G_{l+1}^{(d+l)} \\ A_{l+1}^{(d+l)} \\ B_{l+1}^{(d+l)} \end{bmatrix},$$

$$\begin{bmatrix} F \\ A_F \\ B_F \end{bmatrix} \leftarrow \begin{cases} \begin{bmatrix} G_l^{(d+l-1)} \\ A_l^{(d+l-1)} \\ B_l^{(d+l-1)} \end{bmatrix}, & |\mathrm{lc}(G_l^{(d+l-1)})| \geq |\mathrm{lc}(G_{l+1}^{(d+l-1)})| \\ & \text{または, } \deg G_{l+1}^{(d+l)} < 0 \\ \begin{bmatrix} G_{l+1}^{(d+l-1)} \\ A_{l+1}^{(d+l-1)} \\ B_{l+1}^{(d+l-1)} \end{bmatrix}, & |\mathrm{lc}(G_l^{(d+l-1)})| < |\mathrm{lc}(G_{l+1}^{(d+l-1)})|. \end{cases}$$

1) へ戻る.

手順 3 において, 枢軸多項式の次数が 1 次ずつ減少することを仮定する. このことは, 枢軸多項式の主係数を調べれば確かめることができる.

手順 4 では，多項式 $G_{l+1}^{(d+l)}$ とその主係数の零判定が必要である．零判定は，3.3 節の零判定にしたがう．

本拡張算法において，特に，$l=0$ と固定した方法は拡張改良互除法である．

安定な拡張算法により生成される多項式列は，次の意味を持つ．

定理 3.4.1 本節の安定な拡張算法の手順 4 の過程で得られた $(G_l^{(d+l-1)}, A_l^{(d+l-1)}, B_l^{(d+l-1)})$, $(G_{l+1}^{(d+l)}, A_{l+1}^{(d+l)}, B_{l+1}^{(d+l)})$ について

$$G_l^{(d+l-1)} \sim S_{n-l}(F,G) \tag{3.25}$$
$$G_{l+1}^{(d+l)} \sim S_{n-l-1}(F,G) \tag{3.26}$$

が成り立ち，さらに手順 4 で更新された (F, A_F, B_F), (G, A_G, B_G) について

$$F = A_F F_0 + B_F G_0, \tag{3.27}$$
$$G = A_G F_0 + B_G G_0 \tag{3.28}$$

が成立する．ここで，F_0, G_0 は，本拡張算法の初期設定において与えられる F, G のことである．

[証明] (3.25), (3.26) については，前節の定理 3.3.1 ですでに示した．(3.27), (3.28) を帰納法で示す．最初の段階では，本拡張算法の初期設定から，定理の成立は明らかである．任意の段階で (3.27), (3.28) が成立するとして，次の段階で，それらが成立することを示そう．更新前の多項式の組を (F, A_F, B_F), (G, A_G, B_G) として

$$F = f_m z^m + f_{m-1} z^{m-1} + \cdots + f_0,$$
$$G = g_n z^n + g_{n-1} z^{n-1} + \cdots + g_0$$

とする．手順 1 でパラメータ l に応じて，F, G の $n-l-1$ 次部分終結式

$$S_{n-l-1}(F,G) = \begin{vmatrix} f_m & f_{m-1} & \cdots & \cdots & \cdots & f_{n-2l} & Fz^l \\ & f_m & \cdots & \cdots & \cdots & f_{n-2l+1} & Fz^{l-1} \\ & & \ddots & & & \vdots & \vdots \\ 0 & & & f_m & \cdots & f_{n-l} & Fz^0 \\ g_n & g_{n-1} & \cdots & \cdots & \cdots & g_{n-2l-d} & Gz^{d+l} \\ & g_n & \cdots & \cdots & \cdots & g_{n-2l-d+1} & Gz^{d+l-1} \\ & & \ddots & & & \vdots & \vdots \\ 0 & & & g_n & \cdots & g_{n-l} & Gz^0 \end{vmatrix} \tag{3.29}$$

が定まる．ここで，$d := m-n$ とおいた．帰納法の仮定 (3.27), (3.28) を上式に代入して

て整理すれば

$$S_{n-l-1}(F,G) = \begin{vmatrix} f_m & f_{m-1} & \cdots & \cdots & \cdots & f_{n-2l} & A_F z^l \\ & f_m & \cdots & \cdots & \cdots & f_{n-2l+1} & A_F z^{l-1} \\ & & \ddots & & & \vdots & \vdots \\ 0 & & & f_m & \cdots & f_{n-l} & A_F z^0 \\ g_n & g_{n-1} & \cdots & \cdots & \cdots & g_{n-2l-d} & A_G z^{d+l} \\ & g_n & \cdots & \cdots & \cdots & g_{n-2l-d+1} & A_G z^{d+l-1} \\ & & \ddots & & & \vdots & \vdots \\ 0 & & & g_n & \cdots & g_{n-l} & A_G z^0 \end{vmatrix} F_0$$
$$+ \begin{vmatrix} f_m & f_{m-1} & \cdots & \cdots & \cdots & f_{n-2l} & B_F z^l \\ & f_m & \cdots & \cdots & \cdots & f_{n-2l+1} & B_F z^{l-1} \\ & & \ddots & & & \vdots & \vdots \\ 0 & & & f_m & \cdots & f_{n-l} & B_F z^0 \\ g_n & g_{n-1} & \cdots & \cdots & \cdots & g_{n-2l-d} & B_G z^{d+l} \\ & g_n & \cdots & \cdots & \cdots & g_{n-2l-d+1} & B_G z^{d+l-1} \\ & & \ddots & & & \vdots & \vdots \\ 0 & & & g_n & \cdots & g_{n-l} & B_G z^0 \end{vmatrix} G_0. \tag{3.30}$$

ここで，左辺の行列式および右辺の F_0, G_0 に係る行列式の行列成分は，右端の列成分を除いてすべて等しいことを注意しておく．これらの右端の列成分を同じ行列成分の右側に追加した行列

$$\begin{bmatrix} f_m & f_{m-1} & \cdots & \cdots & \cdots & f_{n-2l} & Fz^l & A_F z^l & B_F z^l \\ & f_m & \cdots & \cdots & \cdots & f_{n-2l+1} & Fz^{l-1} & A_F z^{l-1} & B_F z^{l-1} \\ & & \ddots & & & \vdots & \vdots & \vdots & \vdots \\ 0 & & & f_m & \cdots & f_{n-l} & Fz^0 & A_F z^0 & B_F z^0 \\ g_n & g_{n-1} & \cdots & \cdots & \cdots & g_{n-2l-d} & Gz^{d+l} & A_G z^{d+l} & B_G z^{d+l} \\ & g_n & \cdots & \cdots & \cdots & g_{n-2l-d+1} & Gz^{d+l-1} & A_G z^{d+l-1} & B_G z^{d+l-1} \\ & & \ddots & & & \vdots & \vdots & \vdots & \vdots \\ 0 & & & g_n & \cdots & g_{n-l} & Gz^0 & A_G z^0 & B_G z^0 \end{bmatrix} \tag{3.31}$$

を行の枢軸交換をしながら上三角化する．この上三角化により，(3.30) 式の左辺の行列式および右辺の F_0, G_0 に係る行列式の行列成分が同時に上三角化される．本拡張算法では，この上三角化を多項式演算 Piv と Elim を用いて行っている．上三角化後の行列成分の右下隅の 3 つの要素（右端から 3, 2, 1 番目）に多項式の組 $(G_{l+1}^{(d+l)}, A_{l+1}^{(d+l)}, B_{l+1}^{(d+l)})$ が現れる．等式 (3.30) と 行列 (3.31) の上三角化により

$$G_{l+1}^{(d+l)} = A_{l+1}^{(d+l)} F_0 + B_{l+1}^{(d+l)} G_0 \tag{3.32}$$

が成り立つ．

以上により，多項式の組 $(G_{l+1}^{(d+l)}, A_{l+1}^{(d+l)}, B_{l+1}^{(d+l)})$ を改めて (G, A_G, B_G) と更新すれば，(3.28) を満たす．

多項式の組 (F, A_F, B_F) の更新は，手順 4 により行列 (3.31) の三角化が完了する直前の多項式の 2 つの組

$$(G_l^{(d+l-1)}, A_l^{(d+l-1)}, B_l^{(d+l-1)}),\ (G_{l+1}^{(d+l-1)}, A_{l+1}^{(d+l-1)}, B_{l+1}^{(d+l-1)})$$

の中で，いずれか一方である．等式 (3.32) の場合と同様に，両者ともに (3.27) を満たすことが確かめられる．■

定理 3.4.2 定理 3.4.1 の条件の下で，手順 4 によって更新された多項式 F と G の次数に関して

$$\begin{aligned} \deg G &\leq \deg G_0 - \deg A_G - d, && (3.33)\\ \deg F &\leq \deg G_0 - \deg A_G && (3.34) \end{aligned}$$

が成立する．ただし，$d := \deg F - \deg G \geq 1$.

[証明]　帰納法で証明する．まず，最初の段階で定理の成立を確かめる．

手順 3 の仮定から

$$\deg F = \deg G_0 - l. \tag{3.35}$$

一方，多項式 A_G は (3.30) の右辺の F_0 に係る行列式と相似であり，また初期設定から

$$\deg A_G \leq l \tag{3.36}$$

が成り立つ．したがって

$$\begin{aligned} \deg G &= \deg F - d \\ &= \deg G_0 - l - d \\ &\leq \deg G_0 - \deg A_G - d \end{aligned}$$

となり，(3.33) が成立する．ここで，2 番目の等式は (3.35)，3 番目の不等式は (3.36) を用いた．また，上の結果を用いれば，直ちに

$$\begin{aligned} \deg F &= \deg G + d \\ &\leq \deg G_0 - \deg A_G \end{aligned}$$

が得られ，したがって (3.34) が成立する．よって，最初の段階で定理は成立する．

次に，任意の段階で定理の成立を仮定し，更新された多項式 $\tilde{F}$，$A_{\tilde{F}}$，$B_{\tilde{F}}$ および $\tilde{G}$，$A_{\tilde{G}}$，$B_{\tilde{G}}$ について考える．$\tilde{d} := \deg \tilde{F} - \deg \tilde{G}$ とおく．パラメータ l が $\tilde{l}$ に選択されたとして，再び手順 3 の仮定から

$$\deg \tilde{F} = \deg G - \tilde{l}. \tag{3.37}$$

一方，(3.30) 式の右辺の F_0 に係る行列式において，d, l をそれぞれ $\tilde{d}$, $\tilde{l}$ で置き換えれば

$$\deg A_{\tilde{G}} \leq \deg A_G + \tilde{d} + \tilde{l}. \tag{3.38}$$

したがって，更新された多項式 $\tilde{G}$ の次数は次のように評価される．

$$\begin{aligned} \deg \tilde{G} &= \deg \tilde{F} - \tilde{d} \\ &= \deg G - \tilde{l} - \tilde{d} \\ &\leq (\deg G_0 - \deg A_G - d) - \tilde{l} - \tilde{d} \\ &= \deg G_0 - (\deg A_G + d + \tilde{l}) - \tilde{d} \\ &\leq \deg G_0 - \deg A_{\tilde{G}} - \tilde{d}. \end{aligned}$$

ここで，2 番目の等式には (3.37) 式，3 番目の不等式には帰納法の仮定の不等式 (3.33)，最後の不等式には (3.38) 式を用いた．さらに，上の結果を用いれば

$$\begin{aligned} \deg \tilde{F} &= \deg \tilde{G} + \tilde{d} \\ &\leq \deg G_0 - \deg A_{\tilde{G}} \end{aligned}$$

が得られる．よって，定理が成立する．■

注 1. (3.33), (3.34) の不等式について，実は等号が成立することが後の第 4.1 章の系 4.1.2 で分かる．

注 2. 上の定理において，更新された $\tilde{G}$，$A_{\tilde{G}}$，$B_{\tilde{G}}$ は，本拡張算法における $G_{l+1}^{(d+l)}$，$A_{l+1}^{(d+l)}$，$B_{l+1}^{(d+l)}$ に相等する．

定理 3.4.1 と定理 3.4.2 から，ある部分終結式と相似である多項式 $G_l^{(d+l-1)}$ とそれに付随する多項式 $A_l^{(d+l-1)}$，$B_l^{(d+l-1)}$ について，次の系が得られる．

系 3.4.1 定理 3.4.1 の条件の下で，多項式の組 $(G_l^{(d+l-1)}, A_l^{(d+l-1)}, B_l^{(d+l-1)})$ に関して

$$\begin{aligned} G_l^{(d+l-1)} &= A_l^{(d+l-1)} F_0 + B_l^{(d+l-1)} G_0, \\ \deg G_l^{(d+l-1)} &\leq \deg G_0 - \deg A_l^{(d+l-1)} - 1 \end{aligned}$$

が成立する．

数値例 3.4.1 3.3 節と同じ問題に対して，本拡張算法の計算結果を示す．

本拡張算法の計算結果

$$\begin{aligned} P_1 &\sim -0.85714278839286 - 0.12499992666667z - 8.8888812500000 \times 10^{-2} z^2 \\ &\quad - 6.6666583333325 \times 10^{-2} z^3 + 1.7493686873168 \times 10^{-7} z^4, \\ A_1 &\sim -1.0000000000000. \end{aligned}$$

$$
\begin{aligned}
B_1 &\sim 9.1666666701684 \times 10^{-8} + 1.0000000000000z, \\
P_4 &\sim -0.86574944736120 - 0.23056649210443z, \\
A_4 &\sim -1.0000000000000 - 0.10738169634887z \\
&\quad + 0.13592692995171z^2 + 8.6893907764700 \times 10^{-2}z^3, \\
B_4 &\sim -1.147545362445\underline{6} \times 10^{-2} + 0.9942070997446\underline{0}z + 0.10738169908985z^2 \\
&\quad - 0.13592693791698z^3 - 8.6893907764700 \times 10^{-2}z^4, \\
P_5 &\sim -0.8622302908507\underline{4}, \\
A_5 &\sim -1.0000000000000 + 0.1550390523426\underline{0}z + 8.956786775286\underline{9} \times 10^{-2}z^2 \\
&\quad + 5.7759858539682 \times 10^{-2}z^3 - 2.0350783141947 \times 10^{-2}z^4, \\
B_5 &\sim -6.783244943839\underline{2} \times 10^{-3} + 0.99671463954855z - 0.1536823300050\underline{0}z^2 \\
&\quad - 8.956787660762\underline{5} \times 10^{-2}z^3 - 5.775985667419\underline{4} \times 10^{-2}z^4 \\
&\quad + 2.0350783141947 \times 10^{-2}z^5.
\end{aligned}
$$

これより，最終項の精度は約 13 桁である．前節の拡張改良互除法の結果と比較して数値的安定性の点で改善されている．

3.5 まとめ

浮動小数点演算の下で，古典的な Euclid 算法による多項式剰余列の生成は，数値的に不安定である．特に剰余列生成の過程で不正規に近い状態になったときに，Euclid 算法では隣接する剰余列要素から次の剰余列要素を計算するため桁落ちが生ずる．改良互除法では隣接する剰余列ではなく，次数の高い方の多項式として，主係数の絶対値の大きい方と組み合わせて，次の剰余列要素を求めることにより，桁落ちを抑制することができた．本算法では，この改良互除法を一般化して，パラメータ l を自動的に選択し，比較的安定に求まる剰余列要素だけを計算した．本算法において，パラメータ l を零にとったとき，改良互除法となる．多項式演算として，主係数消去演算 Elim と多項式列の主係数の枢軸選び Piv を定義して，これらを用いて改良互除法および安定な算法を記述した．特に本算法における多項式演算は G.E.Collins が導入した部分終結式を構成する行列の枢軸選び付き上三角化と対応していることを示した．改良互除法および本算法を関数の有理関数補間，孫子定理の応用に利用できるように，これらの拡張算法について述べた．拡張算法は，多項式演算 Elim と Piv を拡張定義するだけで記述できることを示し，有理関数補間で必要となる次数に関する条件を満たしていることを示した．

第 4 章

Givens 回転による多項式剰余列の安定な生成法

多項式を行ベクトル，多項式列を行列に対応させ，多項式剰余列の生成法を線形代数の用語で記述する．我々は，Givens の面回転による行列の QR 分解によって多項式剰余列の安定な算法を提示する．算法の安定性を 2 - ノルムの意味で考える．

2 つの多項式 $F(z)$ と $G(z)$ から生成される多項式剰余列に関連して，後に必要な事項と記号から述べる．

4.1 基本的事項

4.1.1 新しい記号の導入

多項式 $F,\ G \in K[z]$ を

$$
\begin{aligned}
F &= f_m z^m + f_{m-1}z^{m-1} + \cdots + f_0, && (4.1)\\
G &= g_n z^n + g_{n-1}z^{n-1} + \cdots + g_0,\ f_m \neq 0,\ g_n \neq 0,\ n \leq m. && (4.2)
\end{aligned}
$$

として，新たに記号を導入する．

$\|F\|_1, \|F\|_2$: それぞれ $F(z)$ の 1 - ノルム，2 - ノルムとよび，以下のように定義する．

$$\|F\|_1 := \sum_{k=0}^{m} |f_k|,\ \|F\|_2 := (\sum_{k=0}^{m} |f_k|^2)^{1/2}.$$

注意．多項式の積 $F \cdot G$ のノルムについて，$\|F \cdot G\|_2 \leq \|F\|_1 \cdot \|G\|_2$ が成り立つ．

$A \sim B$: 行列 A，B において，一方が他方の左からユニタリ行列を掛けたものと等しいとき，両者を同一視して，記号~を用いる．

$\|A\|_E$: 行列 $A=(a_{ij})$ の Euclid ノルム

$$\|A\|_E := (\sum_{i,j}|a_{ij}|^2)^{1/2}.$$

$L_k(F,G)$: 非負整数 $k \le n$ を与える．便宜上 $d=\deg F-\deg G$ とおき，$2k+d$ 個の多項式系

$$\{F, zF, \ldots, z^{k-1}F, G, zG, \ldots, z^{k+d-1}G\}$$

が張る実または複素線形空間．

上の線形空間の包含関係について，明らかに

$$L_0(F,G) \subset L_1(F,G) \subset \cdots \subset L_n(F,G)$$

が成り立つ．

L_k : 線形空間 $L_k(F,G)$ を張る多項式系を $2k+d$ 個の行ベクトルと同一視して，次のように行列表示する．

$$L_k := \begin{bmatrix} z^{k-1}F \\ \vdots \\ F \\ z^{k+d-1}G \\ \vdots \\ G \end{bmatrix} = \begin{bmatrix} f_m & f_{m-1} & \cdots & f_0 & & \\ & \cdots & \cdots & & & \\ & & f_m & f_{m-1} & \cdots & f_0 \\ g_n & \cdots & g_0 & & & \\ & \cdots & \cdots & & & \\ & & & g_n & \cdots & g_0 \end{bmatrix}.$$

以上を準備して，まず線形空間 $L_k(F,G)$ の次元を定める．

4.1.2 線形空間 $L_k(F,G)$ の次元

定理 4.1.1 線形空間 $L_k(F,G)$ の次元 $\dim L_k(F,G)$ について，次式が成立する．

$$\dim L_k(F,G) = \begin{cases} 2k+d, & k \le \deg G-\mu, \\ k+\deg F-\mu, & k > \deg G-\mu. \end{cases}$$

ここで，$\mu=\deg \gcd(F,G)$.

[証明] 最初に，番号 k が，$0 \le k \le n-\mu$ のときの線形空間の次元が，$2k+d$ であることを示す．これは，$2k+d$ 個の多項式系

$$\{F, zF, \ldots, z^{k-1}F, G, zG, \ldots, z^{k+d-1}G\}$$

が線形独立であることを示せばよい．これらの線形結合について

$$\sum_{r=0}^{k-1} a_r z^r F + \sum_{s=0}^{k+d-1} b_s z^s G = 0$$

となったとする．上式において

$$A := \sum_{r=0}^{k-1} a_r z^r,\ B := \sum_{s=0}^{k+d-1} b_s z^s,\ \tilde{F} := F/\gcd(F,G),\ \tilde{G} := G/\gcd(F,G)$$

とおけば

$$A\tilde{F} + B\tilde{G} = 0$$

となる．ここで，$\tilde{F}$ と $\tilde{G}$ は，互いに素であるので，A は $\tilde{G}$ で割り切れる．一方，番号 k について，$k \le n-\mu$ であるので，$\deg \tilde{G} > \deg A$ となる．したがって，A は零多項式となり，さらに，このとき B も零多項式となる．すなわち

$$a_r = 0\ (0 \le r \le k-1),\ b_s = 0\ (0 \le s \le k+d-1).$$

よって，$2k+d$ 個の多項式系の線形独立性が示された．

つぎに，番号 k が，$n-\mu+1 \le k \le n$ のときの線形空間の次元が，$k+m-\mu$ となることを示す．これは，$k+m-\mu$ 個の多項式系

$$\{F, zF, \ldots, z^{k-1}F, G, zG, \ldots, z^{m-\mu-1}G\}$$

が張る $L_k(F,G)$ の部分空間を S とおいて，$S=L_k(F,G)$ となること，および上の多項式系の線形独立性を示せばよい．

まず，$S=L_k(F,G)$ となることを示す．$S \subset L_k(F,G)$ となることは，明らかであるので，逆の包含関係を示す．これは，$\{z^sG\}_{s=m-\mu}^{k+d-1} \subset S$ を示せばよい．

$m-\mu \le s \le k+d-1$ なる s について，$z^sG \in S$ となることを，s に関する帰納法で示す．

i) $s=m-\mu$ のとき．$\tilde{F}=F/(F,G)$, $\tilde{G}=G/(F,G)$ とすれば

$$\tilde{G}F - \tilde{F}G = (F,G)(\tilde{G}\tilde{F} - \tilde{F}\tilde{G}) = 0.$$

上式において

$$\tilde{G} := \sum_{i=0}^{n-\mu} \tilde{g}_i z^i,\ \tilde{F} := \sum_{j=0}^{m-\mu} \tilde{f}_j z^j$$

とおけば

$$\sum_{i=0}^{n-\mu} \tilde{g}_i z^i F - \sum_{j=0}^{m-\mu} \tilde{f}_j z^j G = 0$$

となる．この式を $z^{m-\mu}G$ について解く．

$$z^{m-\mu}G = \sum_{i=0}^{n-\mu} \frac{\tilde{g}_i}{\tilde{f}_{m-\mu}} z^i F - \sum_{j=0}^{m-\mu-1} \frac{\tilde{f}_j}{\tilde{f}_{m-\mu}} z^j G.$$

上式右辺は S に属するので，よって，$s=m-\mu$ のときは成立する．

ii) 帰納法の仮定として，番号 s が $m-\mu \le s < k+d-1$ のとき，$z^sG \in S$ が成り立っているとする．このとき，$z^{s+1}G \in S$ を示す．上式の両辺に $z^{s-m+\mu+1}$ を掛けて整理すれば

$$z^{s+1}G = \sum_{i=s-m+\mu+1}^{s-m+n+1} \frac{\tilde{g}_{i-s+m-\mu-1}}{\tilde{f}_{m-\mu}} z^i F - \sum_{j=s-m+\mu+1}^{s} \frac{\tilde{f}_{j-s+m-\mu-1}}{\tilde{f}_{m-\mu}} z^j G.$$

上式右辺の第 1 項の和の上限は，k より小である．一方，右辺第 2 項は，帰納法の仮定より S に属する．よって，$z^{s+1}G \in S$ となる．

よって，i)，ii) より，$\{z^sG\}_{s=m-\mu}^{k+d-1} \subset S$ が示された．

つぎに，線形空間 S を張る $k+m-\mu$ 個の多項式系の線形独立性を示す．これらの線形結合について

$$\sum_{r=0}^{k-1} a_r z^r F + \sum_{s=0}^{m-\mu-1} b_s z^s G = 0$$

が成り立っているとすれば，この関係式より

$$\sum_{r=0}^{n-\mu-1} a_r z^r F + \sum_{s=0}^{m-\mu-1} b_s z^s G = -\sum_{r=n-\mu}^{k-1} a_r z^r F$$

となる．ここで，上式左辺の次数は，高々 $m+n-\mu-1$ 次であるので，上式右辺において

$$a_r = 0 \ \ (n-\mu \le r \le k-1)$$

となる．このとき，上式は

$$\sum_{r=0}^{n-\mu-1} a_r z^r F + \sum_{s=0}^{m-\mu-1} b_s z^s G = 0$$

となる．ここで，最初に示した結果において，番号 k が $n-\mu$ の場合，多項式系

$$\{F, zF, \ldots, z^{n-\mu-1}F, G, zG, \ldots, z^{m-\mu-1}G\}$$

は線形独立であるので，上の式から

$$a_r = 0 \ \ (0 \le r \le n-\mu-1),\ b_s = 0 \ \ (0 \le s \le m-\mu-1)$$

となる．よって，$k+m-\mu$ 個の多項式系の線形独立性が示された．■

定理 4.1.1より $L_k(F,G)$ の次元が定まったので，次に線形空間 $L_k(F,G)$ および行列 L_k に関する性質をいくつか挙げる．

4.1.3 線形空間 $L_k(F,G)$ に関する性質

補助定理 4.1.1 (4.1)，(4.2) で与えられる多項式 F，G において，F を G で割った商多項式を Q，剰余多項式を R とし

$$F = QG + R, \quad \deg R < \deg G$$

とする．このとき，$d = \deg F - \deg G$ とおけば

$$L_k(F,G) = \mathrm{span}\{z^rG, z^sR \mid 0 \le r \le k+d-1,\ 0 \le s \le k-1\}, \quad 1 \le k \le n \qquad (4.3)$$

が成り立つ．

[証明]　右辺の線形空間を S とおく．線形空間 $L_k(F,G)$ は，定義から

$$L_k(F,G) = \mathrm{span}\{z^rG, z^sF \mid 0 \le r \le k+d-1,\ 0 \le s \le k-1\}, \quad 1 \le k \le n$$

であるので，$L_k(F,G) \subset S$ を示すためには $\{z^sF\}_{s=0}^{k-1} \subset S$ を示せばよい．逆の包含関係は，$\{z^sR\}_{s=0}^{k-1} \subset L_k(F,G)$ を示せばよい．包含関係 $\{z^sF\}_{s=0}^{k-1} \subset S$ のみを示す．商多項式 Q を

$$Q = \sum_{j=0}^{d} q_j z^j$$

とすれば

$$\begin{aligned} F &= \sum_{j=0}^{d} q_j z^j G + R \\ &\in \mathrm{span}\{z^rG, R \mid 0 \le r \le d\} \end{aligned}$$

より

$$z^jF \in \mathrm{span}\{z^rG, z^jR \mid j \le r \le j+d\}, \quad 0 \le j \le k-1$$

となる．したがって

$$\begin{aligned} z^jF &\in \bigcup_{\nu=0}^{k-1} \mathrm{span}\{z^rG, z^\nu R \mid \nu \le r \le \nu+d\} \\ &\subset \mathrm{span}\{z^rG, z^sR \mid 0 \le r \le k+d-1,\ 0 \le s \le k-1\} \\ &= S, \quad 0 \le j \le k-1 \end{aligned}$$

となり，包含関係 $\{z^sF\}_{s=0}^{k-1} \subset S$ が示される．

他方の包含関係 $\{z^sR\}_{s=0}^{k-1} \subset L_k(F,G)$ もこれと同様にして示される．■

補助定理 4.1.2 (4.1)，(4.2) で与えられる多項式 F，G において，F と G の多項式剰余列を $\{P_i\}_{i=-1}^{t}$ とし，$d_i = \deg P_{i-1} - \deg P_i \ (0 \le i \le t)$ としたときに，$0 \le i \le t$ に対して

$$L_k(P_{i-1}, P_i) = \mathrm{span}\{z^rP_i, z^sP_{i+1} \mid 0 \le r \le k+d_i-1,\ 0 \le s \le k-1\},\ 1 \le k \le n_i \ \ (4.4)$$

が成り立つ．上式において，特に番号 k が $d_{i+1} < k \ (i < t)$ のとき

$$L_k(P_{i-1}, P_i) = L_{k-d_{i+1}}(P_i, P_{i+1}) \cup \mathrm{span}\{z^rP_i \mid k-d_{i+1} \le r \le k+d_i-1\} \qquad (4.5)$$

となる．

[証明]　補助定理 4.1.1 において，F，G，R をそれぞれ P_{i-1}，P_i，P_{i+1} で置き換えれば，(4.4) が得られる．さらに (4.4) 式において，番号 k が $k > d_{i+1}$ $(i < t)$ のとき

$$\begin{aligned}
&L_k(P_{i-1}, P_i)\\
&\quad = \mathrm{span}\{z^r P_i, z^s P_{i+1} \mid 0 \le r \le (k - d_{i+1}) - 1,\ 0 \le s \le (k - d_{i+1}) + d_{i+1} - 1\}\\
&\qquad \cup \mathrm{span}\{z^r P_i \mid k - d_{i+1} \le r \le k + d_i - 1\}\\
&\quad = L_{k-d_{i+1}}(P_i, P_{i+1}) \cup \mathrm{span}\{z^r P_i \mid k - d_{i+1} \le r \le k + d_i - 1\}
\end{aligned}$$

となり，(4.5) 式が得られる．■

定理 4.1.2 補助定理 4.1.2 において，$n_i = \deg P_i$ $(-1 \le i \le t)$ とすれば，線形空間 $L_k(F, G)$ は，多項式剰余列による基底によって次のように構成される．

$$\begin{aligned}
&L_k(F, G) =\\
&\quad \mathrm{span}\{z^r P_i, z^s P_{i+1} \mid 0 \le r \le k - n + n_{i-1} - 1,\ 0 \le s \le k - n + n_i - 1\}\\
&\qquad \bigoplus \left\{ \bigoplus_{j=0}^{i-1} \mathrm{span}\{z^r P_j \mid k - n + n_{j+1} \le r \le k - n + n_{j-1} - 1\} \right\},\\
&\qquad\qquad n - n_i + 1 \le k \le n - n_{i+1} \quad (0 \le i < t). \qquad (4.6)\\
&\quad \mathrm{span}\{z^r P_t \mid 0 \le r \le k - n + n_{t-1} - 1\}\\
&\qquad \bigoplus \left\{ \bigoplus_{j=0}^{t-1} \mathrm{span}\{z^r P_j \mid k - n + n_{j+1} \le r \le k - n + n_{j-1} - 1\} \right\},\\
&\qquad\qquad n - n_t + 1 \le k \le n. \qquad (4.7)
\end{aligned}$$

ただし，直和の上限が負のときは，結果は空集合とする．

[証明]　$\sigma_j = \sum_{r=1}^{j} d_r$ $(0 \le j \le t)$ とおく．ただし，$\sigma_0 = 0$ とする．σ_j の定義より $\sigma_j = n - n_j$ となる．このとき，$n - n_i + 1 \le k$ $(0 \le i \le t)$ なる番号 k に対して

$$n_j - n_i + 1 \le k - \sigma_j,\ 0 \le j \le t$$

となり，特に

$$d_{j+1} < k - \sigma_j,\ (0 \le j < i) \qquad (4.8)$$

となる．上式と補助定理 4.1.2 の (4.5) 式において，$j = 0$ のときは，$d_1 < k$ となるので，

$$\begin{aligned}
L_k(F, G) &= L_k(P_{-1}, P_0)\\
&= L_{k-d_1}(P_0, P_1) \cup \mathrm{span}\{z^r P_0 \mid k - d_1 \le r \le k + d_0 - 1\}\\
&= L_{k-\sigma_1}(P_0, P_1) \cup \mathrm{span}\{z^r P_0 \mid k - n + n_1 \le r \le k - n + n_{-1} - 1\} \qquad (4.9)
\end{aligned}$$

$1 \le j < i$ については

$$\begin{aligned}
&L_{k-\sigma_j}(P_{j-1}, P_j)\\
&\quad = L_{k-\sigma_{j+1}}(P_j, P_{j+1}) \cup \mathrm{span}\{z^r P_j \mid k - \sigma_{j+1} \le r \le k - \sigma_{j-1} - 1\}\\
&\quad = L_{k-\sigma_{j+1}}(P_j, P_{j+1}) \cup \mathrm{span}\{z^r P_j \mid k - n + n_{j+1} \le r \le k - n + n_{j-1} - 1\}
\end{aligned} \qquad (4.10)$$

が成り立つ．上の 2 式より

$$\begin{aligned}
&L_k(F, G)\\
&\quad = L_{k-\sigma_i}(P_{i-1}, P_i) \cup \left\{ \bigcup_{j=0}^{i-1} \mathrm{span}\{z^r P_j \mid k - n + n_{j+1} \le r \le k - n + n_{j-1} - 1\} \right\},\\
&\qquad\qquad n - n_i + 1 \le k\ (0 \le i \le t) \qquad (4.11)
\end{aligned}$$

が得られる．ここで，右辺の j に関する和集合を構成する多項式系の個数は

$$\begin{aligned}
&\sum_{j=0}^{i-1} \{(k - n + n_{j-1} - 1) - (k - n + n_{j+1}) + 1\}\\
&\quad = \sum_{j=0}^{i-1} (n_{j-1} - n_{j+1})\\
&\quad = \sum_{j=0}^{i-1} (n_{j-1} - n_j) + \sum_{j=0}^{i-1} (n_j - n_{j+1})\\
&\quad = (n_{-1} - n_{i-1}) + (n_0 - n_i) = m - n_{i-1} + n - n_i \qquad (4.12)
\end{aligned}$$

となる．(4.11) 式において，番号 k が

$$n - n_i + 1 \le k \le n - n_{i+1}\ (0 \le i < t)$$

のとき

$$1 \le k - \sigma_i \le n_i - n_{i+1}\ (0 \le i < t)$$

となるので，$L_{k-\sigma_i}(P_{i-1}, P_i)$ に対して補助定理 4.1.2 が適用できて

$$\begin{aligned}
&L_{k-\sigma_i}(P_{i-1}, P_i)\\
&\quad = \mathrm{span}\{z^r P_i, z^s P_{i+1} \mid 0 \le r \le (k - \sigma_i) + d_i - 1,\ 0 \le s \le (k - \sigma_i) - 1\}\\
&\quad = \mathrm{span}\{z^r P_i, z^s P_{i+1} \mid 0 \le r \le k - n + n_{i-1} - 1,\ 0 \le s \le k - n + n_i - 1\}
\end{aligned} \qquad (4.13)$$

となる．このとき，$L_k(F, G)$ を構成する多項式系の個数は，(4.12) を用いて

$$(k - n + n_{i-1}) + (k - n + n_i) + (m - n_{i-1} + n - n_i) = 2k + d$$

となる．また，$L_k(F, G)$ の次元は，定理 4.1.1 より $2k + d$ となり，次元の数と多項式系の個数が一致する．したがってこれらの多項式系は，$L_k(F, G)$ の基底となり，(4.6) 式が得られる．

(4.7) 式については，(4.11) 式において，番号 k が

$$n - n_t + 1 \le k \le n$$

のとき

$$1 \le k - \sigma_t \le n_t$$

となるので，(4.13) と同様にして

$$\begin{aligned} & L_{k-\sigma_t}(P_{t-1}, P_t) \\ &= \mathrm{span}\{z^r P_t, z^s P_{t+1} \mid 0 \le r \le (k-\sigma_t) + d_t - 1,\ 0 \le s \le (k-\sigma_t) - 1\} \\ &= \mathrm{span}\{z^r P_t \mid 0 \le r \le k - n + n_{t-1} - 1\} \end{aligned} \tag{4.14}$$

を得る．このとき，$L_k(F,G)$ を構成する多項式系の個数は，(4.12) を用いて

$$(k - n + n_{t-1}) + (m - n_{t-1} + n - n_t) = k + m - \mu$$

となる．また，定理 4.1.1 より，$L_k(F,G)$ の次元は，$k+m-\mu$ であり，多項式系の個数と一致する．したがって，これらの多項式系は $L_k(F,G)$ の基底となり，(4.7) 式が得られる．■

定理 4.1.2 による $L_k(F,G)$ の基底を高次の順に並べれば次のようになる．

$n - n_i + 1 \le k \le n - n_{i+1}\ (0 \le i < t)$.

次数	基底
$k-n+n_{-1}+n_0-1$	$z^{k-n+n_{-1}-1}P_0$
$k-n+n_{-1}+n_0-2$	$z^{k-n+n_{-1}-2}P_0$
$\vdots$	$\vdots$
$k-n+n_1+n_0$	$z^{k-n+n_1}P_0$
$\vdots$	$\vdots$
$k-n+n_{i-1}+n_i-1$	$z^{k-n+n_{i-1}-1}P_i$
$\vdots$	$\vdots$
n_i	P_i
n_i-1	□
$\vdots$	$\vdots$
$n_i+n_{i+1}-k$	□
$k-n+n_i+n_{i+1}-1$	$z^{k-n+n_i-1}P_{i+1}$
$k-n+n_i+n_{i+1}-2$	$z^{k-n+n_i-2}P_{i+1}$
$\vdots$	$\vdots$
n_{i+1}	P_{i+1}

$n - n_t + 1 \le k \le n$.

次数	基底
$k-n+n_{-1}+n_0-1$	$z^{k-n+n_{-1}-1}P_0$
$k-n+n_{-1}+n_0-2$	$z^{k-n+n_{-1}-2}P_0$
$\vdots$	$\vdots$
$k-n+n_1+n_0$	$z^{k-n+n_1}P_0$
$\vdots$	$\vdots$
$k-n+n_{t-1}+n_t-1$	$z^{k-n+n_{t-1}-1}P_t$
$k-n+n_{t-1}+n_t-2$	$z^{k-n+n_{t-1}-2}P_t$
$\vdots$	$\vdots$
n_t	P_t

左上の □ は，左の次数に対応する多項式がないことを表す．□ の個数は $k - n_{i+1}$ 個である．$k = n_{i+1}$ のときは □ は存在しない．

系 4.1.1 定理 4.1.2 の条件のもとで，線形空間 $L_k(F,G)$ の最低次数多項式は，剰余列要素である．特に，番号 k が

$$n - n_i + 1 \le k \le n - n_{i+1}\ (0 \le i < t)$$

のときの最低次数多項式は P_{i+1} であり

$$n - n_t \le k \le n$$

のときの最低次数多項式は P_t である．

系 4.1.2 定理 4.1.2 の条件のもとで，線形空間 $L_k(F,G)\ (0 \le k \le n - \mu)$ において，剰余列要素 P_{i+1} を含む最小の空間は $L_{n-n_i+1}(F,G)$ であり，P_{i+1} を最低次数多項式として含む最大の空間は，$L_{n-n_{i+1}}(F,G)$ である．ただし，$0 \le i < t$ である．

定理 4.1.3 任意の $P \in L_k(F,G)$ について，$\deg P \le n - k$ ならば P は $L_k(F,G)$ の最低次数多項式で割り切れる．

[証明]　系 4.1.1 より，番号 k が $n - n_t \le k \le n$ のとき，$L_k(F,G)$ の最低次数多項式は P_t である．一方，このときの P の次数は高々 n_t であるので，P は P_t と一致するかあるいは零多項式となる．したがって，この場合は定理が成立する．次に，番号 k が $n - n_i + 1 \le k \le n - n_{i+1}\ (0 \le i < t)$ の場合を考える．再び系 4.1.1 より，このときの $L_k(F,G)$ の最低次数多項式は P_{i+1} である．P を P_{i+1} で割った商多項式を Q，剰余多項式を R とすれば

$$R = P - QP_{i+1},\ \deg R < \deg P_{i+1} \tag{4.15}$$

となる．上式において

$$R \in L_{n-n_{i+1}}(F,G) \tag{4.16}$$

が示されれば，P_{i+1} が $L_{n-n_{i+1}}(F,G)$ の最低次数多項式であることから，$R = 0$ となり，定理が得られる．したがって，(4.16) 式を示す．P について，$P \in L_k(F,G)$ より

$$\begin{aligned} & {}^{\exists_1}A,\ B\ :\ \deg A \le k-1,\ \deg B \le k + d - 1 \quad \text{s.t.} \\ & \qquad P = AF + BG. \end{aligned} \tag{4.17}$$

一方，系 4.1.2 より P_{i+1} を含む最小の線形空間は $L_{n-n_i+1}(F,G)$ であるので，P_{i+1} について

$$\begin{aligned} & {}^{\exists_1}A_{i+1},\ B_{i+1}\ :\ \deg A_{i+1} = n - n_i,\ \deg B_{i+1} = m - n_i \quad \text{s.t.} \\ & \qquad P_{i+1} = A_{i+1}F + B_{i+1}G \end{aligned} \tag{4.18}$$

が成り立つ．ここで，(4.17)，(4.18) 式を使って R を F と G で表現すれば

$$R = (A - QA_{i+1})F + (B - QB_{i+1})G$$

となる．上式右辺において

$$\tilde{A} = A - QA_{i+1} \tag{4.19}$$
$$\tilde{B} = B - QB_{i+1} \tag{4.20}$$

とおく．(4.15) 式において，Q の次数は $\deg P \leq n-k$ を用いて

$$\begin{aligned} \deg Q &= \deg P - \deg P_{i+1} \\ &\leq n-k-n_{i+1} \end{aligned}$$

と評価される．このとき，$\tilde{A}$ の次数は (4.19) 式より

$$\begin{aligned} \deg \tilde{A} &\leq \max(\deg A, \deg Q + \deg A_{i+1}) \\ &\leq \max(k-1, (n-k-n_{i+1}) + n - n_i) \\ &\leq \max(k-1, (n_i - n_{i+1} - 1) + n - n_i) \\ &\leq n - n_{i+1} - 1 \end{aligned} \tag{4.21}$$

と評価される．上式 3，4 番目の評価においては，番号 k に関する不等式

$$n - n_i + 1 \leq k \leq n - n_{i+1}$$

を用いた．$\tilde{B}$ の次数も，同様にして

$$\deg \tilde{B} \leq m - n_{i+1} - 1 \tag{4.22}$$

と評価される．(4.21)，(4.22) 式は，(4.16) の成立を意味する．よって，定理が示された．
■

4.1.4 行列 L_k に関する性質

定理 4.1.4 線形空間 $L_k(F,G)$ の行列表現 L_k を左から正則行列を掛けて上三角化したとき，最下位の行ベクトルが零ベクトルでなければ，線形空間の次元は $2k+d$ である．そのとき，最下位の行ベクトルを係数にもつ多項式は，ある剰余列要素の定数倍あるいは多項式倍である．

[証明]　行列 L_k を上三角化した行列において，最下位行を係数にもつ多項式を P とする．このとき P の次数は高々 $n-k$ 次であるので，P が零多項式でなければ定理 4.1.3 より，ある剰余列要素の定数倍あるいは多項式倍となる．次に

$$P \neq 0 \Longrightarrow \dim L_k(F,G) = 2k+d \tag{4.23}$$

を示す．$L_k(F,G)$ の次元が $2k+d$ でないとする．このとき，定理 4.1.1 より番号 k は $k > n-\mu$ となる．したがって $\deg P < \mu$ となり，P は零多項式となる．これは $P \neq 0$ に反する．よって (4.23) が成り立つ．■

定理 4.1.5 系 4.1.2の条件の下で，線形空間 $L_k(F,G)$ の行列表現 L_kにおいて，番号 kが $n-n_i+1$ あるいは $n-n_{i+1}$ であるとき，行列 L_k を上三角化したときの対角要素は，最下位要素を除いてすべて零でない．ここで，番号 i は $0 \leq i < t$ である．特に，番号 k が $n-n_{i+1}$ であるとき，対角要素はすべて零でない．また，上三角化後の $2k+d$ 個の行ベクトルを多項式に対応させて，上の行から順に

$$R_1,\ R_2,\ \ldots,\ R_{2k+d-1},\ R_{2k+d} \tag{4.24}$$

とすれば，多項式列 $\{R_j\}_{j=1}^{2k+d}$ は，線形空間 $L_k(F,G)$ の次数の相異なる基底をなし，その最低次数多項式 R_{2k+d} は，剰余列要素 P_{i+1} の定数倍である．また，それらの次数について

$$\begin{aligned} &\deg R_1 = \deg F + k - 1, \\ &\deg R_j = \deg R_{j-1} - 1,\ 1 < j < 2k+d \end{aligned} \tag{4.25}$$

が成立する．特に $k = n - n_{i+1}$ であれば，$\deg R_{2k+d} = \deg R_{2k+d-1} - 1$ である．

[証明]　定理 4.1.2 より番号 k が $n-n_i+1$ または $n-n_{i+1}\,(0 \leq i \leq t)$ のとき，$L_k(F,G)$ の多項式剰余列による基底を高次の順に

$$S_1,\ S_2,\ \ldots,\ S_{2k+d-1},\ S_{2k+d} \tag{4.26}$$

とすれば，それらの次数は最高次から $\deg S_{2k+d-1}$ まで 1 次ずつ下がる．すなわち

$$\begin{aligned} &\deg S_1 = \deg F + k - 1, \\ &\deg S_j = \deg S_{j-1} - 1,\ 1 < j < 2k+d, \\ &\deg S_{2k+d} \leq \deg S_{2k+d-1} - 1\ (k = n - n_{i+1} \text{ のとき等号成立}). \end{aligned} \tag{4.27}$$

特に，S_{2k+d} は剰余列要素 P_{i+1} である．また行列 L_k を上三角化して得られる多項式系 (4.24) は，$L_k(F,G)$ の新たな基底である．

$$L_k(F,G) = \mathrm{span}\{R_1, \ldots, R_{2k+d-1}, R_{2k+d}\}. \tag{4.28}$$

上式右辺の多項式系の作り方より，それらの次数について

$$\begin{aligned} &\deg R_1 \leq \deg F + k - 1 \\ &\deg R_j \leq \deg F + k - j, 1 < j \leq 2k+d \end{aligned} \tag{4.29}$$

となる．(4.27)，(4.29) 式より

$$\deg R_j \leq \deg S_j,\ 1 \leq j < 2k+d \tag{4.30}$$

となる．上式において，実は等号が成立することをを j に関する帰納法で示す．

i)　$j=1$ のとき，$\deg R_1 < \deg S_1$ とすれば

$$\deg R_j \leq \deg S_j < \deg S_1,\ j > 1$$

より

$$\max_{1\le j\le 2k+d} \deg R_j < \deg S_1$$

となる．一方

$$S_1 \in \mathrm{span}\{R_1, R_2, \ldots, R_{2k+d-1}, R_{2k+d}\}$$

である．上式において，右辺の多項式系の任意の線形結合によって得られる多項式の最高次数は $\deg S_1$ より低いので不合理である．よって $\deg R_1 = \deg S_1$ となる．

ii) 帰納法の仮定として

$$\deg R_r = \deg S_r,\ 1 \le r \le j \tag{4.31}$$

が成り立っているとする．このとき $\deg R_{j+1} = \deg S_{j+1}$ を示す．
$\deg R_{j+1} < \deg S_{j+1}$ として矛盾を導く．

$$\deg R_r \le \deg S_r < \deg S_{j+1},\ r > j+1$$

より

$$\max_{j+1\le r\le 2k+d} \deg R_r < \deg S_{j+1}.$$

一方，帰納法の仮定より

$$S_{j+1} \in \mathrm{span}\{R_{j+1}, \ldots, R_{2k+d}\}$$

となる．前述と同様の議論により，これは不合理．よって，$\deg R_{j+1} = \deg S_{j+1}$ が成り立つ．

よって，i), ii) より

$$\deg R_j = \deg S_j,\ 1 \le j < 2k+d$$

となるので，(4.25) が得られる．また

$$\deg R_{2k+d} \le \deg R_{2k+d-1} - 1$$

より，S_{2k+d} は $L_k(F,G)$ の最低次数多項式となり，剰余列要素 P_{i+1} と一致する．■

系 4.1.3 行列 L_k，L_{k+1} $(k < n,\ n = \deg G)$ を上三角化した行列をそれぞれ T_k，T_{k+1} とし，行列 T_k，T_{k+1} の最下位の行ベクトルをそれぞれ $\boldsymbol{t}_k$，$\boldsymbol{t}_{k+1}$ とする．このとき，$\boldsymbol{t}_k \neq 0$ かつ $\boldsymbol{t}_{k+1} = 0$ ならば，行ベクトル $\boldsymbol{t}_k$ の要素を係数にもつ多項式は，F と G の最大公約因子である．さらに行列 T_k は，零対角要素を持たない上三角行列であって，この行列の $2k+d$ 個の行ベクトルに対応する多項式列は，線形空間 $L_k(F,G)$ の次数の相異なる基底をなし，それらの次数は，最高次から最大公約因子の次数まで 1 次ずつ下がる．

4.2 算法

Givens 回転による多項式剰余列の算法を行列の記法にしたがって説明する．そのため，必要に応じて一つの多項式を一つの行ベクトルに対応させる．

4.2.1 Givens 回転による線形空間 $L_k(F,G)$ の 基底の帰納的構成法

実または複素係数の多項式 P を，算法上

$$P = p_m z^m + p_{m-1} z^{m-1} + \cdots + p_0$$

と表現する．$p_m \neq 0$ は仮定せず，Pの形式的次数 m を真の次数 $\deg P$ と区別するため，$\mathrm{Deg}P$ と表記する．明らかに $\deg P \le \mathrm{Deg}P$ である．この不等式の等号成立は，$p_m \neq 0$ のときであり，このとき，形式次数は真の次数と一致する．

式 (4.1)，(4.2) で与えられた多項式 F，G において，$L_k(F,G)$ の基底として，形式的次数の相異なる多項式系を導出することは，行列 L_k を正則変換により上三角化することである．前章では，この上三角化を部分枢軸選択と消去法を用いて行った．本章では，最も簡単な直交変換である Givens 回転を用いる．これによって，算法の数値的安定性が，さらに改善されるだけでなく，後述する F と G の剰余列の拡張算法において，ある種の直交関係が成り立つ（後述の定理 4.3.1）．

形式的次数の等しい 2 つの多項式

$$\begin{aligned} P &= p_m z^m + p_{m-1} z^{m-1} + \cdots + p_0, \\ Q &= q_m z^m + q_{m-1} z^{m-1} + \cdots + q_0 \end{aligned}$$

を 2 つの行ベクトルとみて

$$\begin{pmatrix} P \\ Q \end{pmatrix} = \begin{pmatrix} p_m & p_{m-1} & \cdots & p_0 \\ q_m & q_{m-1} & \cdots & q_0 \end{pmatrix}$$

と書き，Q の主係数 q_m を Givens 回転によって消去し，Q の形式的次数 $\mathrm{Deg}Q$ を 1 次だけ下げる．

$$\mathrm{Givens}\begin{pmatrix} P \\ Q \end{pmatrix} = \begin{pmatrix} \bar{c} & -\bar{s} \\ s & c \end{pmatrix}\begin{pmatrix} P \\ Q \end{pmatrix}. \tag{4.32}$$

$$c = \frac{p_m}{\sqrt{|p_m|^2 + |q_m|^2}},\ s = \frac{-q_m}{\sqrt{|p_m|^2 + |q_m|^2}},\ q_m \neq 0.$$

$q_m = 0$ のとき，演算子 Givens は恒等変換とする．ここで，$\bar{c}$ は c の共役複素数．

番号 k を $1, 2, \ldots$ と順次動かしながら，$L_k(F,G)$ の形式的次数の相異なる基底を帰納的に構成する．

まず $k = 1$ の場合である．

$$L_1 = \begin{bmatrix} F \\ z^d G \\ \vdots \\ z^0 G \end{bmatrix} = \begin{bmatrix} f_m & f_{m-1} & \cdots & \cdots & f_0 \\ g_n & \cdots & g_0 & & \\ & \ddots & & \ddots & \\ 0 & & g_n & \cdots & g_0 \end{bmatrix}$$

すなわち上 Hessenberg 行列 L_1 を QR 分解により上三角化する．この上三角化を後の都合を考えて，二段階に分けて扱う．

第一段階として，L_1 の上 2 行に Givens 回転

$$\begin{pmatrix} F^{(1)} \\ G^{(1)} \end{pmatrix} = \mathrm{Givens} \begin{pmatrix} F \\ z^d G \end{pmatrix} \tag{4.33}$$

を施せば

$$L_1 \sim \begin{bmatrix} F^{(1)} \\ G^{(1)} \\ z^{d-1}G \\ \vdots \\ z^0 G \end{bmatrix} = \begin{bmatrix} * & * & \cdots & \cdots & \cdots & * \\ & * & \cdots & \cdots & \cdots & * \\ & * & \cdots & \cdots & \cdots & * \\ & & \ddots & & & \vdots \\ 0 & & & * & \cdots & * \end{bmatrix}.$$

第二段階として，上式右辺の 1 行 1 列を除いた上 Hessenberg 行列

$$\begin{bmatrix} G^{(1)} \\ z^{d-1}G \\ \vdots \\ z^0 G \end{bmatrix} = \begin{bmatrix} * & \cdots & \cdots & \cdots & * \\ * & \cdots & \cdots & \cdots & * \\ & \ddots & & & \vdots \\ 0 & & * & \cdots & * \end{bmatrix}$$

に Givens 回転を d 回施して上三角化する：
$W = G^{(1)}$ とおき

$$\begin{pmatrix} H_i^{(1)} \\ W \end{pmatrix} = \mathrm{Givens} \begin{pmatrix} z^{d-i}G \\ W \end{pmatrix},\ i = 1, \ldots, d. \tag{4.34}$$

ここで等号は，プログラム記法の代入に相当する．この反復計算で得られた W を $H_{d+1}^{(1)}$ とおけば，L_1 は直交変換によって

$$L_1 \sim \begin{bmatrix} F^{(1)} \\ H_1^{(1)} \\ \vdots \\ H_{d+1}^{(1)} \end{bmatrix} = \begin{bmatrix} * & * & \cdots & \cdots & \cdots & * \\ & * & \cdots & \cdots & \cdots & * \\ & & \ddots & & & \vdots \\ 0 & & & * & \cdots & * \end{bmatrix} \tag{4.35}$$

と変換されたことになる．上式右辺の多項式系の構成法より，これらの形式的次数は，

$$\begin{aligned} \mathrm{Deg} F^{(1)} &= m, \\ \mathrm{Deg} H_i^{(1)} &= m - i, \quad i = 1, 2, \ldots, d+1 \end{aligned}$$

となり，行列 L_1 は（広義）上三角化されている．このとき，$H_{d+1}^{(1)} = 0$ ならば，系 4.1.3 より，$L_0(F, G)$ の最低次数多項式 G は，F と G の最大公約因子となる．$H_{d+1}^{(1)} \neq 0$ ならば，定理 4.1.4 より，式 (4.35) の右辺の多項式系は，$L_1(F, G)$ の形式的次数の相異なる基底をなし，さらに，$k = 1$ のとき，$k = n - n + 1 = n - n_0 + 1$ であるので，定理 4.1.5 より $H_{d+1}^{(1)}$ を除いた多項式系の形式的次数は，真の次数と一致し，$H_{d+1}^{(1)}$ は剰余列要素 P_1 の定数倍となる．

帰納法の仮定として，$L_k (k \geq 2)$ が次の漸化式

$$\left[\begin{pmatrix} F^{(1)} \\ G^{(1)} \end{pmatrix} = \mathrm{Givens} \begin{pmatrix} F \\ z^d G \end{pmatrix}, \right.$$
$$\left. \begin{pmatrix} F^{(i+1)} \\ G^{(i+1)} \end{pmatrix} = \mathrm{Givens} \begin{pmatrix} F^{(i)} \\ z G^{(i)} \end{pmatrix}, \quad i = 1, 2, \ldots \right. \tag{4.36}$$

で定義された $F^{(i)}$ によって，次のように上三角化され，その最下位行は，$H_{k+d}^{(k)} \neq 0$ とする．

$$L_k \sim \begin{bmatrix} z^{k-1}F^{(1)} \\ z^{k-2}F^{(2)} \\ \vdots \\ F^{(k)} \\ H_1^{(k)} \\ H_2^{(k)} \\ \vdots \\ H_{k+d}^{(k)} \end{bmatrix} = \begin{bmatrix} * & * & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & * \\ & * & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & * \\ & & \ddots & & & & & & & \vdots \\ & & & * & * & * & \cdots & \cdots & \cdots & * \\ & & & & * & * & \cdots & \cdots & \cdots & * \\ & & & & & * & \cdots & \cdots & \cdots & * \\ & & & & & & \ddots & & & \vdots \\ 0 & & & & & & & * & \cdots & * \end{bmatrix}. \tag{4.37}$$

そこで，L_{k+1} の上三角化について考える．

$$L_{k+1} \sim \begin{bmatrix} z^k F \\ z^{k+d} G \\ L_k \end{bmatrix} \sim \begin{bmatrix} z^k F \\ z^{k+d} G \\ z^{k-1} F^{(1)} \\ \vdots \\ F^{(k)} \\ H_1^{(k)} \\ \vdots \\ H_{k+d}^{(k)} \end{bmatrix} = \begin{bmatrix} * & * & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & * \\ * & * & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & * \\ & * & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & * \\ & & \ddots & & & & & & \vdots \\ & & & * & * & \cdots & \cdots & \cdots & * \\ & & & & * & \cdots & \cdots & \cdots & * \\ & & & & & \ddots & & & \vdots \\ & & & & & & * & \cdots & * \end{bmatrix}.$$

ここで，上式右辺は，帰納法の仮定により上 Hessenberg 行列である．この行列に Givens 回転を施す．まず，上 2 行に Givens 回転を施せば，式 (4.36) より

$$\mathrm{Givens} \begin{pmatrix} z^k F \\ z^{k+d} G \end{pmatrix} = z^k \times \mathrm{Givens} \begin{pmatrix} F \\ z^d G \end{pmatrix} = z^k \times \begin{pmatrix} F^{(1)} \\ G^{(1)} \end{pmatrix}.$$

以下順に

$$\mathrm{Givens} \begin{pmatrix} z^{k-i} F \\ z^{k+1-i} G^{(i)} \end{pmatrix} = z^{k-i} \times \mathrm{Givens} \begin{pmatrix} F^{(i)} \\ z G^{(i)} \end{pmatrix} = z^{k-i} \times \begin{pmatrix} F^{(i+1)} \\ G^{(i+1)} \end{pmatrix},\ i = 1, 2, \ldots, k.$$

したがって

$$L_{k+1}\sim\begin{bmatrix} z^kF^{(1)}\\ \vdots\\ F^{(k+1)}\\ G^{(k+1)}\\ H_1^{(k)}\\ \vdots\\ H_{k+d}^{(k)}\end{bmatrix}=\begin{bmatrix} *&\cdots&\cdots&\cdots&\cdots&\cdots&\cdots&*\\ &\ddots&&&&&&\vdots\\ &&*&*&\cdots&\cdots&\cdots&*\\ &&&*&\cdots&\cdots&\cdots&*\\ &&&*&\cdots&\cdots&\cdots&*\\ &&&&\ddots&&&\vdots\\ &&&&&*&\cdots&*\end{bmatrix}.$$

次に，第二段階操作

$$\begin{bmatrix} W=G^{(k+1)},\\ \begin{pmatrix}H_i^{(k+1)}\\ W\end{pmatrix}=\mathrm{Givens}\begin{pmatrix}H_i^{(k)}\\ W\end{pmatrix},\quad i=1,2,\ldots,k+d,\\ H_{k+d+1}^{(k+1)}=W\end{bmatrix}\tag{4.38}$$

によって，上式の $k+2$ 行目以降の上 Hessenberg 行列は上三角化される．したがって

$$L_{k+1}\sim\begin{bmatrix} z^kF^{(1)}\\ \vdots\\ F^{(k+1)}\\ H_1^{(k+1)}\\ \vdots\\ H_{k+d+1}^{(k+1)}\end{bmatrix}=\begin{bmatrix} *&\cdots&\cdots&\cdots&\cdots&\cdots&\cdots&*\\ &\ddots&&&&&&\vdots\\ &&*&*&\cdots&\cdots&\cdots&*\\ &&&*&\cdots&\cdots&\cdots&*\\ &&&&\ddots&&&\vdots\\ &&&&&*&\cdots&*\end{bmatrix}\tag{4.39}$$

が得られる．ここで，$H_{k+d+1}^{(k+1)}=0$ ならば，L_k を上三角化して得られた式 (4.37) の右辺の多項式系は，系 4.1.3 より，すべて真の次数と一致し，特に，最低次数多項式 $H_{k+d}^{(k)}$ は，F と G の最大公約因子となる．$H_{k+d+1}^{(k+1)}\neq 0$ ならば，L_{k+1} を上三角化して得られた式 (4.39) の右辺の多項式系は，定理 4.1.4 より，線形空間 $L_{k+1}(F,G)$ の形式的次数の相異なる基底をなし，多項式系の構成法から，これらの形式的次数は

$$\begin{aligned}\mathrm{Deg}F^{(i)} &= m,\quad i=1,\ldots,k+1\\ \mathrm{Deg}H_i^{(k+1)} &= m-i,\quad i=1,2,\ldots,k+d+1\end{aligned}$$

となる．特に，番号 $k+1$ が $n-n_i+1$ または，$n-n_{i+1}$ ならば，定理 4.1.5 より $H_{k+d+1}^{(k+1)}$ を除いた多項式系の形式的次数は，真の次数と一致し，$H_{k+d+1}^{(k+1)}$ は剰余列要素 P_{i+1} の定数倍となる．$k+1=n-n_{i+1}$のときは，$H_{k+d+1}^{(k+1)}$ の形式的次数も真の次数と一致する．

Givens 回転による多項式剰余列の算法を記述すれば，次のようになる．

4.2.2 算法

[アルゴリズム A]

入力： 多項式 F，G $(\deg F\geq\deg G)$.

初期化： $m=\deg F$, $n=\deg G$, $d=m-n\geq 0$.
$Q_{-1}=F$, $Q_0=G$.

$$\begin{pmatrix}F^{(1)}\\ G^{(1)}\end{pmatrix}=\mathrm{Givens}\begin{pmatrix}Q_{-1}\\ z^dQ_0\end{pmatrix}.$$

$W=G^{(1)}$とおき

$$\begin{bmatrix} i=1,2,\ldots,d\\ \begin{pmatrix}H_i^{(1)}\\ W\end{pmatrix}=\mathrm{Givens}\begin{pmatrix}z^{d-i}Q_0\\ W\end{pmatrix}.\end{bmatrix}$$

$W=0$ ならば $l=0$ として停止.
$H_{d+1}^{(1)}=W$, $Q_1=W$.

$$\begin{bmatrix} k=1,2,\ldots,n-1\\ \begin{pmatrix}F^{(k+1)}\\ G^{(k+1)}\end{pmatrix}=\mathrm{Givens}\begin{pmatrix}F^{(k)}\\ zG^{(k)}\end{pmatrix}.\\ W=G^{(k+1)}\text{とおく}.\\ \begin{bmatrix} i=1,2,\ldots,k+d\\ \begin{pmatrix}H_i^{(k+1)}\\ W\end{pmatrix}=\mathrm{Givens}\begin{pmatrix}H_i^{(k)}\\ W\end{pmatrix}.\end{bmatrix}\\ W=0\text{ ならば }l=k\text{として停止}.\\ H_{k+d+1}^{(k+1)}=W,\ Q_{k+1}=W.\end{bmatrix}$$

反復計算：

$l=n$ として停止.

アルゴリズム A で得られた番号 l および三種類の多項式列 $\{Q_i\}$, $\{F^{(i)}\}$, $\{H_i^{(i)}\}$ の意味について述べる．

1. 番号 l の意味.
番号 l について，定理 4.1.1 より，次式が成り立つ．

$$l=n-\mu,\ \mu=\deg\gcd(F,G).\tag{4.40}$$

したがって，系 4.1.2より，線形空間 $L_k(F,G)$ $(1\leq k\leq n-\mu)$ において，$L_l(F,G)$ は，F と G の最大公約因子 $\gcd(F,G)$ を最低次数多項式として含む最大の空間となる．

2. 多項式列 $\{Q_i\}$ の意味.
多項式列 $\{Q_i\}$ に対して，次の選出法を適用すれば，F と G の多項式剰余列が得られる．

[剰余列の選出法]

初期化: $P_{-1}=Q_{-1}$, $P_0=Q_0$, $k=0$, $r=1$.

1) $r>l$ ならば $t=k$ として停止.
$r\le l$ ならば，以下続行.
$k=k+1$
$P_k=Q_r$
$r=n-\deg P_k+1$
1) へ.

定理 4.2.1 上の選出法で得られた部分列 $\{P_k\}_{k=-1}^{t}$ は，F と G の多項式剰余列である．

[証明] P_{-1}, P_0 が剰余列要素であることは，定義より明らか．番号 $k\,(k\ge 1)$ について，P_k が剰余列要素であることを k に関する帰納法で示す.
まず，$k=1$ のとき，$L_1(F,G)$ の要素 Q_1 は，その構成法と定理 4.1.5 より剰余列要素 P_1 の定数倍である.
帰納法の仮定として，P_k が剰余列要素であるとする．このとき，$r=n-\deg P_k+1$ として，Q_r が剰余列要素 P_{k+1} となることを示す．$L_r(F,G)$ の要素である Q_r について，その構成法および r の定義から，定理 4.1.5 を適用すれば，Q_r は，剰余列要素 P_{k+1} の定数倍となる.
また，番号 t は $n-\deg P_t+1>n-\mu$ を満たしている．すなわち

$$\mu\le\deg P_t<\mu+1.$$

したがって，$\deg P_t=\mu$ となり，$\{P_k\}_{k=-1}^{t}$ は，F と G の多項式剰余列となる．■

注意．剰余列の選出法で選出されなかった多項式は，定理 4.1.3 より，剰余列要素で割り切れる多項式である.

3. 多項式列 $\{F^{(i)}\}$ と $\{H_i^{(i)}\}$ の意味.
多項式列 $\{F^{(i)}\}$ と $\{H_i^{(i)}\}$ から

$$\begin{aligned} R_i &= H_{l+d-i+1}^{(i)}, \quad 1\le i\le l+d \\ R_{l+d+i} &= z^{i-1}F^{(l-i+1)}, \quad 1\le i\le l \end{aligned}$$

として，多項式系 $\{R_k\}_{k=1}^{2l+d}$ を定義する．このとき，$R_1=\gcd(F,G)$ であることを注意しておく．上の多項式系は，定理 4.1.4 より，形式的次数の相異なる基底である．さらに，式 (4.40) と定理 4.1.5 より，R_k の次数に関して，次の定理が成立する.

定理 4.2.2 多項式系 $\{R_k\}_{k=1}^{2l+d}$ において，各多項式の形式的次数と真の次数は一致し，それらの次数は

$$\deg R_k=\mu+k-1, \quad 1\le k\le 2l+d.$$

次に計算量を評価する．$\deg F=\deg G=n$ の場合を調べる．便宜上，実係数多項式とする.

k 次多項式の Givens 回転，k 回，$1\le k\le n$.

ところで，2 次元ベクトルに対する Givens 変換において，平方根 1 回，乗算 4 回であるので，全体として

平方根 約 $n^2/2$ 回，乗算 約 $4/3n^3$ 回.

アルゴリズム A で得られた F，G の多項式剰余列は，線形空間 $L_l(F,G)$ の次数の相異なる基底を直交変換によって求める際の中間結果として得られることに注意しておく.

4.3 拡張算法

式 (4.1),(4.2) で与えられた多項式 F，G に対して，$d=m-n$，$\mu=\deg\gcd(F,G)$ として，線形空間 $L_k(F,G)\,(1\le k\le n-\mu)$ の任意の要素 P は多項式系

$$\{F,zF,\ldots,z^{k-1}F\}\cup\{G,zG,\ldots,z^{k+d-1}G\}$$

の線形結合である．したがって，この多項式 P は，高々 $k-1$ 次多項式 A と高々 $k+d-1$ 次の多項式 B で

$$P=AF+BG \tag{4.41}$$

と書ける．定理 4.1.1 より，上の多項式系は 1 次独立であるので，P の A，B による表現は一意的である．このとき，P に A，B を付加した (P,A,B) を多項式 P の拡張表現とよぶ．特に，P が剰余列要素であるとき，拡張剰余列要素とよぶ．この拡張剰余列要素については，定理 2.2.2 の次数に関する条件

$$\deg P\le\deg G-\deg A-1$$

を満たしている.

剰余列の拡張算法とは，拡張剰余列 (P_i,A_i,B_i) を求めることである．前節で述べた算法では，$L_k(F,G)$ を張る多項式系に Givens 回転を施して剰余列を求めた．これに対して本拡張算法では，$L_k(F,G)$ を張る拡張表現による多項式系に Givens 回転を施して拡張剰余列を求める．すなわち，多項式系を構成する各多項式 z^rF，z^sG はそれぞれ

$$z^rF=z^rF+0G,\ z^sG=0F+z^sG$$

と一意的に書けるので，z^rF，z^sG の拡張表現はそれぞれ $(z^rF, z^r, 0)$，$(z^sG, 0, z^s)$ となる．これらの拡張表現による多項式系に対して Givens 回転を施す．

前節の式 (4.32) で定義した形式的次数の等しい多項式 P，Q に作用する演算子 Givens を P，Q の拡張表現 (P, A_P, B_P)，(Q, A_Q, B_Q) に対して拡張定義する．

$$\begin{aligned}\mathrm{Givens}\begin{bmatrix}(P,A_P,B_P)\\(Q,A_Q,B_Q)\end{bmatrix} &= \begin{pmatrix}\bar{c} & -\bar{s}\\ s & c\end{pmatrix}\begin{bmatrix}(P,A_P,B_P)\\(Q,A_Q,B_Q)\end{bmatrix}\\ &= \begin{bmatrix}(\bar{c}P-\bar{s}Q, \bar{c}A_P-\bar{s}A_Q, \bar{c}B_P-\bar{s}B_Q)\\(sP+cQ, sA_P+cA_Q, sB_P+cB_Q)\end{bmatrix}.\end{aligned} \tag{4.42}$$

上式右辺の 2 つの多項式の 3 つ組は，ともに式 (4.41) の関係を満たしており，それぞれ $\bar{c}P-\bar{s}Q$，$sP+cQ$ の拡張表現になっている．したがって，上式で拡張定義された演算子 Givens は，P，Q の拡張表現に対して，P，Q に Givens 回転を施した $\bar{c}P-\bar{s}Q$，$sP+cQ$ の拡張表現に変換する線形写像である．

$L_k(F,G)$ を張る拡張表現による多項式系を $2k+d$ 個の行ベクトルに対応させて，行列表示する．

$$\begin{bmatrix} z^{k-1}F & \vdots & z^{k-1} & 0\\ \vdots & \vdots & \vdots & \vdots\\ z^0F & \vdots & z^0 & 0\\ z^{k+d-1}G & \vdots & 0 & z^{k+d-1}\\ \vdots & \vdots & \vdots & \vdots\\ z^0G & \vdots & 0 & z^0\end{bmatrix} = \begin{bmatrix} f_m & \cdots & f_0 & & \vdots & I_k & 0\\ & \ddots & & \ddots & \vdots & & \\ & & f_m & \cdots & f_0 & \vdots & \\ g_n & \cdots & g_0 & & \vdots & & \\ & \ddots & & \ddots & \vdots & & \\ & & g_n & & g_0 & \vdots & 0 \quad I_{k+d}\end{bmatrix}$$

$$= (L_k, I_{2k+d}).$$

(I_k，I_{k+d}，I_{2k+d} : k，$k+d$，$2k+d$ 次単位行列)

上式において，特に，$k=n-\mu$ のとき，$L_{n-\mu}$ の拡張行列に対して，左から直交変換を施して，上三角化すれば，$L_{n-\mu}(F,G)$ の次数の相異なる拡張表現による基底が得られ，計算の中間結果として F，G の拡張剰余列が得られる．

F と G の剰余列の拡張算法

[アルゴリズム B]

入力 : 多項式 F，G $(\deg F \geq \deg G)$.

初期化 : $m=\deg F$, $n=\deg G$, $d=m-n\geq 0$.

$U_{-1}=(F,1,0)$, $U_0=(G,0,1)$.

$\begin{pmatrix}V_1\\ S\end{pmatrix} = \mathrm{Givens}\begin{pmatrix}U_{-1}\\ z^dU_0\end{pmatrix}$.

$T=S$とおく．

$\left[\begin{array}{l} i=1,2,\ldots,d\\ \begin{pmatrix}W_i^{(1)}\\ T\end{pmatrix} = \mathrm{Givens}\begin{pmatrix}z^{d-i}U_0\\ T\end{pmatrix}.\end{array}\right.$

$Q=0$ ならば $l=0$ として停止．ここで,Q は Tの第 1 成分の多項式.

$W_{d+1}^{(1)}=T$, $U_1=T$.

反復計算 :

$\left[\begin{array}{l} k=1,2,\ldots,n-1\\ \begin{pmatrix}V_{k+1}\\ S\end{pmatrix} = \mathrm{Givens}\begin{pmatrix}V_k\\ zS\end{pmatrix} \ : \ S\text{の更新.}\\ T=S\text{とおく.}\\ \left[\begin{array}{l} i=1,2,\ldots,k+d\\ \begin{pmatrix}W_i^{(k+1)}\\ T\end{pmatrix} = \mathrm{Givens}\begin{pmatrix}W_i^{(k)}\\ T\end{pmatrix}.\end{array}\right.\\ Q=0 \text{ ならば } l=k \text{として停止．ここで,} Q \text{ は } T\text{の第 1 成分の多項式.}\\ W_{k+d+1}^{(k+1)}=T,\ U_{k+1}=T.\end{array}\right.$

$l=n$ として停止.

出力，l，U_i，V_i，$W_i^{(l)}$ は次の意味をもつ．

番号 l は，前節で述べたように $l=n-\mu$ $(\mu=\deg\gcd(F,G))$ である．三種類の多項式列 $\{U_i\}$，$\{V_i\}$，$\{W_i^{(l)}\}$ について，それぞれの第 1 成分を $U_i=(Q_i,\cdot,\cdot)$，$V_i=(F^{(i)},\cdot,\cdot)$，$W_i^{(l)}=(H_i^{(l)},\cdot,\cdot)$ と書くことにする．$\{Q_i\}$ に対して，前節の剰余列の選出法を適用すれば，$\{U_i\}$ の部分列として F と G の拡張剰余列が得られる．$\{V_i\}$，$\{W_i^{(l)}\}$ から作られる多項式系

$$\left\{z^{l-i}V_i \,:\, i=1,2,\ldots,l\right\}\cup\left\{W_i^{(l)} \,:\, i=1,2,\ldots,l+d\right\}$$

は，$L_l(F,G)$ の次数の相異なる拡張表現による基底である．これらをあらためて

$$(R_k,A_k,B_k),\ k=1,2,\ldots,r,\ r=2l+d$$

とおく．A_k，B_kは，それぞれ高々 $l-1$ 次，$l+d-1$ 次の多項式である．これを l 次元，$l+d$ 次元の行ベクトルとみて，両者を (A_k,B_k) と並べて，$2l+d$ 次元の行ベクトルとする．

行列

$$\begin{bmatrix} A_1 & B_1 \\ A_2 & B_2 \\ \vdots & \vdots \\ A_r & B_r \end{bmatrix} = \begin{bmatrix} * & \cdots & * & * & \cdots & * \\ * & \cdots & * & * & \cdots & * \\ \vdots & & \vdots & \vdots & & \vdots \\ * & \cdots & * & * & \cdots & * \end{bmatrix}, \ r = 2l + d \tag{4.43}$$

を定義し，我々が提示した拡張算法の特長を定理として与える．

定理 4.3.1 式 (4.43) で定義される長方形行列は行直交行列である．すなわち

$$A_i A_j^H + B_i B_j^H = \begin{cases} 1 & i = j \\ 0 & i \neq j. \end{cases} \quad \text{ただし，} H \text{ は Hermite 転置．}$$

この証明は，次の簡単な行列に関する補助定理から明らかである．

補助定理 4.3.1 $s \geq r$ として，A を $(r \times s)$ 型の正規化された行直交行列とする．このとき，行列 A に左から直交変換を施しても，行直交性は不変である．

剰余列に対しては，上述の意味の直交性は成立しないが，安定性解析の上で有用な次の結果が成り立つ．

定理 4.3.2 アルゴリズム B で得られた $F(z)$，$G(z)$ の任意の拡張剰余列要素を (P, A, B) とするとき

$$\|A\|_2^2 + \|B\|_2^2 = 1$$

がつねに成立する．

4.4 安定性解析

拡張剰余列要素 (P, A, B) は，次数条件

$$\deg P \leq \deg G - \deg A - 1$$

の下で，式 (4.41) の関係を満たしているが，これは，(P, A, B) の係数に関する線形方程式であり，解 (P, A, B) の誤差評価は，その係数行列の条件数の計算が必要であるので，実用上困難である．それに対して，残差の評価は比較的容易である．残差ノルムが十分小さければ，得られた解が有益な情報を与えることが多い．したがって，我々は，拡張剰余列要素の精度を残差の意味で評価することにする．また，その結果を用いた多項式の零判定および多項式の係数の零判定について述べる．前者は，アルゴリズム A，B の停止条件，後者は，剰余多項式の真の次数を決定するために用いる．以下の誤差解析においては，簡単のため，実係数多項式を取り扱う．

拡張剰余列要素 (P, A, B) を同次方程式

$$P - AF - BG = 0 \tag{4.44}$$

の解と考える．P，A，B の丸め誤差をそれぞれ ΔP，ΔA，ΔB としたときに残差 r は

$$r = (P + \Delta P) - (A + \Delta A)F - (B + \Delta B)G \tag{4.45}$$

である．式 (4.44) をこの式に代入する．

$$r = \Delta P - \Delta A \cdot F - \Delta B \cdot G. \tag{4.46}$$

式 (4.45) で与えられた残差 r を 2 - ノルムで評価する．任意の多項式 F，G の積に関して

$\|F \cdot G\|_2 \leq \|F\|_2 \cdot \|G\|_1$ が成り立っているので，

$$\|r\|_2 \leq \|\Delta P\|_2 + \|\Delta A\|_2 \|F\|_1 + \|\Delta B\|_2 \|G\|_1.$$

ここで，$\|\Delta P\|_2 \approx \|P\|_2 \cdot \varepsilon_M$，$\|\Delta A\|_2 \approx \|A\|_2 \cdot \varepsilon_M$，$\|\Delta B\|_2 \approx \|B\|_2 \cdot \varepsilon_M$
（ε_Mは計算機イプシロン）のとき

$$\|r\|_2 \leq (\|P\|_2 + \|A\|_2 \|F\|_1 + \|B\|_2 \|G\|_1) \cdot \varepsilon_M$$

となる．他方 $P = AF + BG$ であるので

$$\begin{aligned} \|P\|_2 &\leq \|A\|_2 \|F\|_1 + \|B\|_2 \|G\|_1 \\ &\leq (\|A\|_2^2 + \|B\|_2^2)^{1/2} \cdot (\|F\|_1^2 + \|G\|_1^2)^{1/2} \end{aligned} \tag{4.47}$$

が成り立つ．上述の $\|r\|_2$ と $\|P\|_2$ の評価式より

$$\frac{\|r\|_2}{(\|A\|_2^2 + \|B\|_2^2)^{1/2} \cdot (\|F\|_1^2 + \|G\|_1^2)^{1/2}} \leq 2 \cdot \varepsilon_M \tag{4.48}$$

が得られる．上式左辺のように正規化した残差ノルムが $O(\varepsilon_M)$ ならば，拡張剰余列要素 (P, A, B) の精度は，残差の意味で，限界に達していることになる．

アルゴリズム B で得られる拡張剰余列要素 (P, A, B) の精度を計るために，その残差 r を J. H. Wilkinson の誤差解析にしたがって評価する（[38]，pp.131-138）．

行列 L_k に $2k + d$ 次単位行列 I_{2k+d} を付加して (L_k, I_{2k+d}) とする．この行列の行ベクトルを次のように並べ換える．

$$(L_k, I_{2k+d})$$

$$\sim \begin{bmatrix} z^{k-1}F & z^{k-1} & 0 \\ z^{k+d-1}G & 0 & z^{k+d-1} \\ \vdots & \vdots & \vdots \\ F & 1 & 0 \\ z^d G & 0 & z^d \\ z^{d-1}G & 0 & z^{d-1} \\ \vdots & \vdots & \vdots \\ G & 0 & 1 \end{bmatrix} = \begin{bmatrix} * & * & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & * \\ * & * & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & * \\ & & \ddots & & & & & & \vdots \\ & & & * & * & \cdots & \cdots & \cdots & * \\ & & & * & * & \cdots & \cdots & \cdots & * \\ & & & & * & \cdots & \cdots & \cdots & * \\ & & & & & \ddots & & & \vdots \\ 0 & & & & & & * & \cdots & * \end{bmatrix}.$$

上式右辺の行列を $L_k^{(0)}$ とする．アルゴリズム B において，$L_k(F,G)$ の形式的次数の相異なる拡張表現による基底を生成する過程は，行列 $L_k^{(0)}$ に対して，右下隅に現われる上 Hessenberg 小行列の Givens 回転による上三角化を k 回施して上三角化することである．上三角化されるまでに要する Givens 回転の総数は $k\cdot(k+d)$ 回である．上三角化後の各行ベクトルには，$L_k(F,G)$ の形式的次数の相異なる拡張表現による基底が対応している．Givens 回転による上三角化を数値的に行い，その過程で発生する丸め誤差を次の漸化式で定義する．

$$L_k^{(i)} = R_i L_k^{(i-1)} + E_i \quad (i=1,2,\ldots,N_k), \quad N_k = k\cdot(k+d). \tag{4.49}$$

上式において，R_i は，$L_k^{(i-1)}$ に施す正確な Givens 回転の行列であり，$L_k^{(i)}$ は，R_i を数値的に求めた行列を数値的に $L_k^{(i-1)}$ に掛けて得られた行列であり，E_i は，そのときに発生した丸め誤差である．$L_k^{(N_k)}$ は，数値的に得られた上三角行列である．

$L_k^{(0)}$ に数値的な Givens 回転を N_k 回施したときに発生する丸め誤差 Δ_k を

$$\Delta_k = L_k^{(N_k)} - R_{N_k}R_{N_k-1}\cdots R_1 L_k^{(0)} \tag{4.50}$$

と定義し，行列 $L_k^{(N_k)}$ のある行ベクトルが表す拡張表現 (P,A,B) の残差

$$r = P - AF - BG$$

を評価する．この数値的に求めた (P,A,B) の丸め誤差 $(\delta_P,\delta_A,\delta_B)$ は，式 (4.50) で定義した行列 Δ_k の行ベクトル成分である．このとき，残差 r は

$$r = \delta_P - \delta_A F - \delta_B G \tag{4.51}$$

と書ける．丸め誤差行列 Δ_k を (Δ_P,Δ_{AB}) と区分けする．ただし，行列 Δ_P，Δ_{AB} の列の大きさは，それぞれ行列 L_k，I_{2k+d} の列の大きさと一致するようにとる．このとき δ_P は，行列 $L_k^{(0)}(\sim(L_k,I_{2k+d}))$ の L_k 成分に数値的な Givens 回転を施したときの丸め誤差 Δ_P の行ベクトル成分であるので，付録の式 (4.71) を使えば

$$\|\delta_P\|_2 \le 3\cdot(2k+d)^{3/2}\cdot(1+6\cdot\varepsilon_M)^{2k+d}\cdot\|L_k\|_E\cdot\varepsilon_M. \tag{4.52}$$

ここで，上式右辺の $\|L_k\|_E$ は，L_k の定義より

$$\begin{aligned}\|L_k\|_E &= (k\cdot\|F\|_2^2+(k+d)\cdot\|G\|_2^2)^{1/2}\\ &\le \sqrt{k+d}\cdot(\|F\|_2^2+\|G\|_2^2)^{1/2}\\ &\le \sqrt{k+d}\cdot(\|F\|_1^2+\|G\|_1^2)^{1/2}\end{aligned} \tag{4.53}$$

と評価される．

(δ_A,δ_B) は，行列 $L_k^{(0)}(\sim(L_k,I_{2k+d}))$ の I_{2k+d} 成分に数値的な Givens 回転を施したときの丸め誤差 Δ_{AB} の行ベクトル成分であるので，δ_P の評価と同様にして

$$(\|\delta_A\|_2^2+\|\delta_B\|_2^2)^{1/2} \le 3\cdot(2k+d)^{3/2}\cdot(1+6\cdot\varepsilon_M)^{2k+d}\cdot\|I_{2k+d}\|_E\cdot\varepsilon_M \tag{4.54}$$

が成り立つ．

δ_P，(δ_A,δ_B) に関する評価を用いて，式 (4.48) の意味で正規化された残差ノルムを評価する．定理 4.3.2 より $\|A\|_2^2+\|B\|_2^2=1$ であるので，正規化された残差は

$$r/(\|F\|_1^2+\|G\|_1^2)^{1/2}$$

となる．

式 (4.51) によって表された残差 r の 2 - ノルムは δ_P，(δ_A,δ_B) の評価式より

$$\begin{aligned}\|r\|_2 &\le \|\delta_P\|_2+\|\delta_A\|_2\|F\|_1+\|\delta_B\|_2\|G\|_1\\ &\le 3\cdot(2k+d)^{3/2}\cdot(1+6\cdot\varepsilon_M)^{2k+d}\cdot(\|L_k\|_E+\|I_{2k+d}\|_E\cdot(\|F\|_1^2+\|G\|_1^2)^{1/2})\cdot\varepsilon_M.\end{aligned}$$

これを正規化すれば，不等式 (4.53) および $\|I_{2k+d}\|_E=\sqrt{2k+d}$ より

$$\begin{aligned}\frac{\|r\|_2}{(\|F\|_1^2+\|G\|_1^2)^{1/2}} &\le 3\cdot(2k+d)^{3/2}\cdot(1+6\cdot\varepsilon_M)^{2k+d}\cdot(\sqrt{k+d}+\sqrt{2k+d})\cdot\varepsilon_M\\ &\le 6\cdot(2k+d)^2\cdot(1+6\cdot\varepsilon_M)^{2k+d}\cdot\varepsilon_M\end{aligned} \tag{4.55}$$

が得られる．F の次数に上限を設定すれば，上式右辺は $O(\varepsilon_M)$ となり，このときの拡張剰余列要素 (P,A,B) の精度は，残差の意味で，限界に達している．

Euclid 算法において，多項式の零判定は，困難な問題である．しかしながら，本拡張算法において，残差を基準にすれば，零判定は比較的容易である．アルゴリズム B で生成される多項式 P の拡張表現 (P,A,B) において，多項式 P の零判定を行う．記号の便宜上，$\gamma=(\|F\|_1^2+\|G\|_1^2)^{1/2}$ とおいて，多項式 P について

$$\|P\|_2/\gamma = O(\varepsilon_M) \tag{4.56}$$

が成り立つとき，次の評価式

$$\begin{aligned}\|AF+BG\|_2/\gamma &= \|P-r\|_2/\gamma\\ &\le \|P\|_2/\gamma+\|r\|_2/\gamma = O(\varepsilon_M)\end{aligned}$$

が成立する．ここで，上式の最後の評価では，式 (4.55), (4.56) を用いた．したがって，多項式 P が，評価式 (4.56) を満たすとき，$P=0$ とみなして正規化した残差ノルムは，$O(\varepsilon_M)$ となるので，P は零多項式と判定できる．アルゴリズム A，B の停止条件では，この判定法を用いる．

多項式 P の係数の零判定も同様にして，γ との比が，$O(\varepsilon_M)$ となる係数は零とみなす．この原則は，多項式 P の真の次数を決定するために用いる．

4.5 数値例

多項式 F と G の拡張剰余列 $U:=(P,A,B)$ をアルゴリズム B によって，単精度演算 $(\varepsilon_M=2^{-23}\approx 0.12\times10^{-6})$ で計算する．(P,A,B) の残差 (4.45) は，式 (4.48) にしたがって正規化して，その残差を 2 - ノルムで計る．

表 4.1: 拡張 Euclid 算法と Algorithm B との比較

m	ε	残差ノルムの最大値（拡張 Euclid 算法）	残差ノルムの最大値（アルゴリズム B）
3	0.001	0.62×10^{-1}	0.35×10^{-7}
4	0.01	0.84×10^{-1}	0.43×10^{-8}
5	0.1	0.97×10^{-3}	0.97×10^{-8}
10	0.1	精度なし	0.23×10^{-7}
20	0.1	精度なし	0.36×10^{-7}

例 (1). 互いに素である多項式

$$F = z^m + 1,\ G = \varepsilon z^2 + 2z + 1.\quad (m \geq 3,\ \varepsilon > 0)$$

に対して，古典的な拡張 Euclid 算法と比較する．

拡張剰余列の残差ノルムの最大値を比較した結果を 表 4.1 に示す．この問題において，パラメータ m を大きく，ε を小さくとったときに，拡張 Euclid 算法を適用した結果では，剰余列要素 P_2 に付随する多項式 A_2，B_2 のノルムが同時に小さくなり，このとき，P_2，B_2 は桁落ちを起こしている．そのため，P_2 の拡張表現の残差ノルムが大きくなり，精度が悪化している．それに対して，アルゴリズム B を適用した結果では，残差ノルムの最大値が $O(\varepsilon_M)$ であるので，すべての拡張剰余列要素が，限界の精度で得られている．

例 (2). 次の多項式について，2 次の GCD を求める．

$$\begin{aligned} F &= (\sum_{k=0}^{m-2} \alpha_{k+1}^2 \cdot z^k)(z^2 + 0.1z + 0.2), \\ G &= (\sum_{k=0}^{n-2} (-1)^i \alpha_{k+1} \cdot z^k)(z^2 + 0.1z + 0.2). \end{aligned}$$

ここで，数列 $\{\alpha_k\}$ は，区間 $(0,1)$ 上に分布するビット反転列であって，次の漸化式で与える．

$$\begin{cases} \alpha_1 = 1/2, \\ \alpha_{2k} = \alpha_k/2, \\ \alpha_{2k+1} = (1 + \alpha_k)/2,\ k \geq 1. \end{cases}$$

例 (3). 次の多項式について，10 次の GCD を求める．

$$\begin{aligned} F &= (\sum_{k=0}^{90} \alpha_{k+1}/(k+1) \cdot z^k)(\sum_{l=0}^{10} z^l), \\ G &= (\sum_{k=0}^{80} \alpha_{k+1} \cdot z^k)(\sum_{l=0}^{10} z^l). \end{aligned}$$

表 4.2: 例 (2) の結果

(m,n)	残差ノルムの最大値	$\gcd(F,G)$
$(50,25)$	0.60×10^{-7}	$z^2 + 0.1000000z + 0.2000000$
$(100,50)$	0.50×10^{-7}	$z^2 + 9.9999987 \times 10^{-2}z + 0.2000000$
$(200,25)$	0.32×10^{-7}	$z^2 + 9.9999972 \times 10^{-2}z + 0.2000001$

表 4.3: 例 (3) の結果

残差ノルムの最大値$= 0.53 \times 10^{-7}$

$\gcd(F,G) = z^{10} + 1.0000013z^9 + 1.0000000z^8$
$+0.9999993z^7 + 0.9999976z^6 + 1.0000045z^5 + 0.9999971z^4$
$+0.9999980z^3 + 1.0000002z^2 + 1.0000005z + 0.9999962$

例 (2), (3) において，F と G の係数は倍精度で計算してから単精度に丸める．残差ノルムの最大値と主係数を 1 にした GCD を 表 4.2，表 4.3 に示す．単精度演算の拡張 Euclid 算法では，いずれの問題も，桁落ちあるいはオーバーフローのため，GCD が得られなかった．これに対して，アルゴリズム B では，拡張剰余列が限界の精度で得られている．

4.6 まとめ

この章のはじめに述べたように，我々は，1 変数多項式 $F(z)$，$G(z)$ から作られる多項式の全体

$$A(z)F(z) + B(z)G(z),\quad \deg A < \deg G,\quad \deg B < \deg F$$

を線形空間と考え，ここで最も簡単な直交変換である Givens の面回転を用い，次数の相異なる基底

$$R_k(z) = A_k(z)F(z) + B_k(z)G(z)$$

を構成した．$\{R_k(z)\}$ を生成する中間結果として $F(z)$，$G(z)$ の剰余列 $\{P_i(z)\}$ が得られる．

本方法は，多項式を行ベクトルとみたときに，直交変換の繰り返しによる行列の QR 分解であるので，数値的に安定である．第 4.4 節では，その安定性解析を行った．

基底と剰余列の拡張表現において，定理 4.3.1 と定理 4.3.2 が成り立つ．

定理 4.3.2 は，浮動小数点演算の下で，多項式が，恒等的に零か否かを判定するときに合理的な基準を与えてくれる．これに基づき，2 つの多項式の近似 GCD が，限界の精度で求まることを数値例によって確かめた．

なお，$F(z)$ と $G(z)$ の次数がともに n のとき，計算量は n^3 に比例することを注意しておく．

4.7 付録

第 4.4 節の不等式 (4.52)，(4.54) を証明するために，行列に関する必要な定理を与える．$m \geq n$ として，$(n \times m)$ 型実行列 A を Givens 回転によって上三角化したときの丸め誤差を評価する．行列 A の (i,j) - 成分

$$(1,2)(1,3)\ldots(1,n);(2,3)(2,4)\ldots(2,n);\ldots;(n-1,n)$$

を順次消去して上三角化したときの丸め誤差解析は，Wilkinson がすでに行った（[38]，pp.135-138）．これに対して，我々は，アルゴリズム B で得られた拡張剰余列の丸め誤差評価に関連して

$$\begin{matrix} 2k\left\{\begin{matrix}\\ \\ \\ \\ \\ \end{matrix}\right. \\ d\left\{\begin{matrix}\\ \\ \\ \end{matrix}\right. \end{matrix}
\begin{bmatrix}
* & * & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & * \\
* & * & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & * \\
 & & \ddots & & & & & & \vdots \\
 & & & * & * & \cdots & \cdots & \cdots & * \\
 & & & * & * & \cdots & \cdots & \cdots & * \\
 & & & & * & \cdots & \cdots & \cdots & * \\
 & & & & & \ddots & & & \vdots \\
\mathbf{0} & & & & & & * & \cdots & *
\end{bmatrix} \tag{4.57}$$

の形をした行列 A に対して，その (i,j) - 成分

$$\begin{aligned}
&(2k,k)(2k+1,k+1)\ldots(2k+d,k+d);\\
&(2k-2,k-1)(2k-1,k)\ldots(2k+d,k+d+1);\\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots;\\
&(2,1)(3,2)(4,3)\ldots(2k+d-1,2k+d-2)(2k+d,2k+d-1)
\end{aligned} \tag{4.58}$$

を順次消去して A を上三角化したときの丸め誤差評価を行う．

まず，(4.57) の形状をした行列 A において，$d=1$ の場合を考える．

2 次元ベクトル $\boldsymbol{u}$ に正確な Givens 回転の行列 R を数値的に求めたものを施して，数値的に得られた結果を $\boldsymbol{v}$ とし，そのときに発生する丸め誤差を $\boldsymbol{e}$ とする．

$$\boldsymbol{v} = R\boldsymbol{u} + \boldsymbol{e}. \tag{4.59}$$

上式において，丸め誤差 $\boldsymbol{e}$ は

$$\|\boldsymbol{e}\|_2 \leq 6 \cdot \|\boldsymbol{u}\|_2 \cdot \varepsilon_M \tag{4.60}$$

と評価され，さらに

$$\|\boldsymbol{v}\|_2 \leq (1 + 6 \cdot \varepsilon_M) \cdot \|\boldsymbol{u}\|_2 \tag{4.61}$$

が成立する（[38]，p.133）．行列 A の任意の列ベクトル成分 $\boldsymbol{a}$ を

$$\boldsymbol{a} = (a_1, a_2, \ldots, a_{2k}, a_{2k+1})^T$$

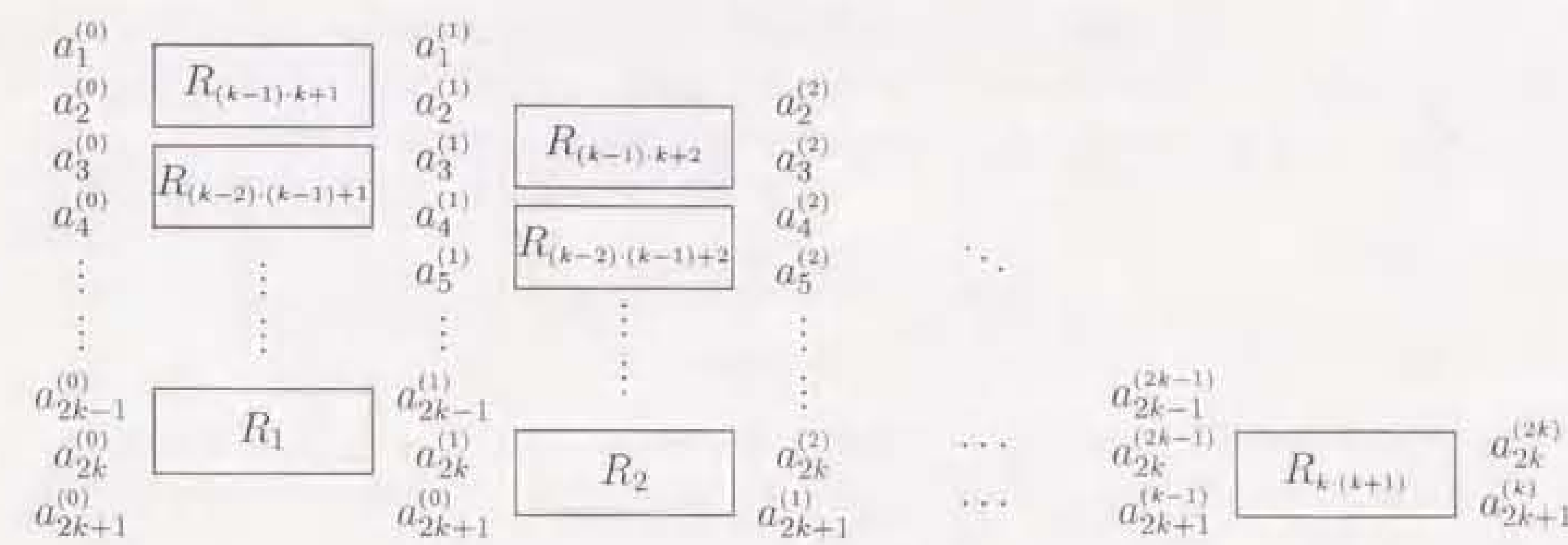

図 4.1: The process of Givens rotation

とする．$a_i^{(0)} = a_i$，$i = 1, 2, \ldots, 2k+1$ と初期化して，ベクトル $\boldsymbol{a}$ に $N = k \cdot (k+1)$ 回の Givens 回転を作用させたときの経過を図 4.1 に示す．

図 4.1 において，縦に並ぶ Givens 回転による関係式を上から順に記述する．

$$\left\{\begin{array}{l}
\begin{pmatrix} a_i^{(i)} \\ a_{i+1}^{(i)} \end{pmatrix} = R_{(k-1)\cdot k+i} \begin{pmatrix} a_i^{(i-1)} \\ a_{i+1}^{(i-1)} \end{pmatrix} + \boldsymbol{e}_{(k-1)\cdot k+i} \\
\begin{pmatrix} a_{i+2}^{(i)} \\ a_{i+3}^{(i)} \end{pmatrix} = R_{(k-2)\cdot(k-1)+i} \begin{pmatrix} a_{i+2}^{(i-1)} \\ a_{i+3}^{(i-1)} \end{pmatrix} + \boldsymbol{e}_{(k-2)\cdot(k-1)+i} \\
\vdots \\
\begin{pmatrix} a_{i+2k-2\cdot[\frac{i+1}{2}]}^{(i)} \\ a_{i+2k-2\cdot[\frac{i+1}{2}]+1}^{(i)} \end{pmatrix} = R_{([\frac{i+1}{2}]-1)\cdot[\frac{i+1}{2}]+i} \begin{pmatrix} a_{i+2k-2\cdot[\frac{i+1}{2}]}^{(i-1)} \\ a_{i+2k-2\cdot[\frac{i+1}{2}]+1}^{(i-1)} \end{pmatrix} + \boldsymbol{e}_{([\frac{i+1}{2}]-1)\cdot[\frac{i+1}{2}]+i} \\
\text{ただし，最後の式において，} i \text{ が偶数のとき} \\
a_{i+2k-2\cdot[\frac{i+1}{2}]+1}^{(i-1)} = a_{2k+1}^{(i/2-1)}, \quad a_{i+2k-2\cdot[\frac{i+1}{2}]+1}^{(i)} = a_{2k+1}^{(i/2)}, \quad 1 \leq i \leq 2k.
\end{array}\right. \tag{4.62}$$

ベクトル $\boldsymbol{a}$ に，$N = k \cdot (k+1)$ 回の Givens 回転を施して，最終的に得られたベクトルを $\boldsymbol{a}^{(N)}$，そのときの丸め誤差 δ を

$$\delta = \boldsymbol{a}^{(N)} - R_N R_{N-1} \cdots R_1 \boldsymbol{a} \tag{4.63}$$

としたときに，丸め誤差 δ について

$$\|\delta\|_2 \leq \sum_{i=1}^{N} \|\boldsymbol{e}_i\|_2 \tag{4.64}$$

が成立する（[38]，p.134）．我々は，上式の右辺をさらに評価する．

式 (4.62) において，式 (4.60)，(4.61) を使えば，次の評価式が得られる．

$$\begin{cases} \sum\limits_{j=1}^{k-[\frac{i+1}{2}]+1} \|\boldsymbol{e}_{(k-j)\cdot(k-j+1)+i}\|_2^2 \le (6\cdot\varepsilon_M)^2 \cdot \sum\limits_{j=1}^{2\cdot(k-[\frac{i+1}{2}]+1)} (a_{i+j-1}^{(i-1)})^2, \\ \sum\limits_{j=1}^{2\cdot(k-[\frac{i+1}{2}]+1)} (a_{i+j-1}^{(i)})^2 \le (1+6\cdot\varepsilon_M)^2 \cdot \sum\limits_{j=1}^{2\cdot(k-[\frac{i+1}{2}]+1)} (a_{i+j-1}^{(i-1)})^2,\ 1\le i\le 2k. \end{cases} \tag{4.65}$$

ここで

$$\alpha_i = \sum_{j=1}^{2k-i+1} (a_{i+j-1}^{(i-1)})^2 + (a_{2k+1}^{([\frac{i-1}{2}])})^2 \tag{4.66}$$

とおけば，式 (4.65) は

$$\begin{cases} \sum\limits_{j=1}^{k-[\frac{i+1}{2}]+1} \|\boldsymbol{e}_{(k-j)\cdot(k-j+1)+i}\|_2^2 \le (6\cdot\varepsilon_M)^2\cdot\alpha_i, \\ (a_i^{(i)})^2 + \alpha_{i+1} \le (1+6\cdot\varepsilon_M)^2\cdot\alpha_i,\ 1\le i\le 2k \end{cases} \tag{4.67}$$

となり，特に，2 番目の式は

$$\alpha_{i+1} \le (1+6\cdot\varepsilon_M)^2\cdot\alpha_i \tag{4.68}$$

となる．また，$\alpha_1 = \|\boldsymbol{a}\|_2^2$ であるので，この漸化式を使えば，α_i は

$$\alpha_i \le (1+6\cdot\varepsilon_M)^{2\cdot(i-1)}\cdot\|\boldsymbol{a}\|_2^2$$

と評価される．この式を式 (4.67) の最初の式に代入する．

$$\sum_{j=1}^{k-[\frac{i+1}{2}]+1} \|\boldsymbol{e}_{(k-j)\cdot(k-j+1)+i}\|_2^2 \le (6\cdot\varepsilon_M)^2\cdot(1+6\cdot\varepsilon_M)^{2\cdot(i-1)}\cdot\|\boldsymbol{a}\|_2^2, 1\le i\le 2k.$$

上の評価式を使えば

$$\begin{aligned} \sum_{i=1}^{k\cdot(k+1)} \|\boldsymbol{e}_i\|_2^2 &= \sum_{i=1}^{2k}\sum_{j=1}^{k-[\frac{i+1}{2}]+1} \|\boldsymbol{e}_{(k-j)\cdot(k-j+1)+i}\|_2^2 \\ &\le 2k\cdot(6\cdot\varepsilon_M)^2\cdot(1+6\cdot\varepsilon_M)^{2\cdot 2k}\cdot\|\boldsymbol{a}\|_2^2. \end{aligned} \tag{4.69}$$

よって，$d=1$ の場合の丸め誤差 δ は

$$\begin{aligned} \|\delta\|_2 &\le \sum_{i=1}^{k\cdot(k+1)} \|\boldsymbol{e}_i\|_2 \\ &\le \sqrt{k\cdot(k+1)}\cdot(\sum_{i=1}^{k\cdot(k+1)} \|\boldsymbol{e}_i\|_2^2)^{1/2} \\ &\le (2k+1)/2\cdot\sqrt{2k}\cdot 6\cdot(1+6\cdot\varepsilon_M)^{2k}\cdot\|\boldsymbol{a}\|_2\cdot\varepsilon_M \\ &\le 3\cdot(2k+1)^{3/2}\cdot(1+6\cdot\varepsilon_M)^{2k+1}\cdot\|\boldsymbol{a}\|_2\cdot\varepsilon_M \end{aligned}$$

と評価される．

$d=0$ の場合の丸め誤差 δ の評価については，行列 A の最下行に零ベクトルを追加して，$d=1$ の場合で考える．行列 A の任意の列ベクトル成分 $\boldsymbol{a}$ およびこのベクトルの最下位に零要素を加えたベクトルを $\tilde{\boldsymbol{a}}$ とする．

$$\begin{aligned} \boldsymbol{a} &= (a_1, a_2, \ldots, a_{2k})^T, \\ \tilde{\boldsymbol{a}} &= (a_1, a_2, \ldots, a_{2k}, 0)^T. \end{aligned}$$

ベクトル $\tilde{\boldsymbol{a}}$ に $N = k\cdot(k+1)$ 回 Givens 回転して得られるベクトルを $\tilde{\boldsymbol{a}}^{(N)}$ とする．このとき，図 4.1 において，$a_{2k+1}^{(i)} = 0,\ i = 1, 2, \ldots, k$ であるので

$$\begin{cases} R_{i\cdot(i+1)} = I_2, \\ \boldsymbol{e}_{i\cdot(i+1)} = 0,\ 1\le i\le k \end{cases} \tag{4.70}$$

となる．ここで I_2 は 2 次の単位行列．

上式より，Givens 回転数 $k\cdot(k+1)$ 回から，k 回の恒等変換を差し引いた k^2 回が，実質の Givens 回転数となる．ここで，$\tilde{\boldsymbol{a}}^{(N)}$ の最下位の要素を除いたベクトルは，行列 A に，この k^2 回の Givens 回転を施して上三角化したときのベクトル $\boldsymbol{a}^{(k^2)}$ と一致していることに注意する．また，$\boldsymbol{e}_i$ の総数は，k^2 個であるので，不等式

$$\sum_{i=1}^{k\cdot(k+1)} \|\boldsymbol{e}_i\|_2 \le k\cdot(\sum_{i=1}^{k\cdot(k+1)} \|\boldsymbol{e}_i\|_2^2)^{1/2}$$

が成立する．この不等式および式 (4.69) より，$d=0$ の場合の丸め誤差 δ は

$$\|\delta\|_2 \le 3\cdot(2k)^{3/2}\cdot(1+6\cdot\varepsilon_M)^{2k}\cdot\|\boldsymbol{a}\|_2\cdot\varepsilon_M$$

と評価される．

一般に，$d>1$ の場合

$$2k+d = \begin{cases} 2(k+\tilde{k})+1, & d = 2\tilde{k}+1 \\ 2(k+\tilde{k}), & d = 2\tilde{k} \end{cases}$$

であるので，$k+\tilde{k}$ をあらためて k とすれば，$d=1$ または 0 の場合に帰着される．

以上を定理としてまとめる．

定理 4.7.1 形状 (4.57) の実行列 A の (i,j) - 成分を式 (4.58) の順に Givens 回転によって消去して，上三角化したとき，行列 A の任意の列ベクトル成分 $\boldsymbol{a}$ に対して，式 (4.63) で与えられる丸め誤差 δ は，

$$\|\delta\|_2 \le 3\cdot(2k+d)^{3/2}\cdot(1+6\cdot\varepsilon_M)^{2k+d}\cdot\|\boldsymbol{a}\|_2\cdot\varepsilon_M \tag{4.71}$$

と評価される．

第 4.4 節の不等式 (4.52)，(4.54) は，行列 $L_k^{(0)}$ およびその任意の列ベクトル成分 $\boldsymbol{v}$ に対して，上の定理を適用すれば，ただちに得られる．

第 5 章

有理関数補間への応用

関数 $f(z)$ の有理関数補間問題は関数の零点，極の探索，あるいは制御工学において，インパルス応答から伝達関数を求める問題すなわち Z 変換と関連しており，応用上重要である．本章では，多項式 $X(z)$ の零点上における関数 $f(z)$ の補間 $f[X]$ およびその差分商 $f[X;z]$ を複素積分表示することにより，複素領域上における有理補間の誤差評価を行う．原点を中心としたある閉円板上で有理型である関数を取り扱う．

5.1 記号の定義

複素平面上において，原点を中心とした半径 ρ の開円板 D_ρ および閉円板 $\overline{D}_\rho$ をそれぞれ

$$\begin{aligned} D_\rho &:= \{z \,|\, |z| < \rho\}, \\ \overline{D}_\rho &:= \{z \,|\, |z| \le \rho\} \end{aligned}$$

で定義する．

原点において解析的である関数 $f(z)$ の原点まわりの級数展開

$$f(z) = \sum_{j=0}^{\infty} f_j z^j$$

に対して，次の部分和を記号で定義する．

$$[f(z)]_m^n := \sum_{j=m}^{n} f_j z^j$$

解析関数 $f(z)$ と多項式 $X(z)$ において，$f(z)$ が $X(z)$ の零点で正則であるとき，$f(z)$ の $X(z)$ の零点上における補間多項式，およびその差分商が定義できる．それぞれ記号で $f[X]$，$f[X;z]$ と書く．このとき $f(z)$ は，これらを用いて

$$f(z) = f[X] + f[X;z]X, \ \deg f[X] < \deg X$$

と一意に表される．特に $f(z)$ が多項式のとき，$f[X]$，$f[X;z]$ は，それぞれ $f(z)$ を $X(z)$ で割ったときの剰余多項式および商多項式となる．次の $f[X]$，$f[X;z]$ の複素積分表示は，多項式補間の誤差評価に用いられる．

5.2 補間多項式および差分商の複素積分表示

関数 $f(z)$ は複素平面上の単連結領域 D で解析的であり，多項式 $X(z)$ は，D の内部に零点を持つものとする．

定理 5.2.1 [3, 16] 単連結領域 D で解析的な関数 $f(z)$ および多項式 $X(z)$ において，$f[X]$，$f[X;z]$ の複素積分表示は

$$f[X] = \frac{1}{2\pi i}\oint_C \left(1 - \frac{X(z)}{X(\zeta)}\right)\frac{f(\zeta)}{\zeta - z}d\zeta, \tag{5.1}$$

$$f[X;z] = \frac{1}{2\pi i}\oint_C \frac{f(\zeta)}{X(\zeta)(\zeta - z)}d\zeta \tag{5.2}$$

で与えられる．ここで，C は D の内部の単純閉曲線であって，C で囲まれた領域内に $X(z)$ の零点は存在するものとする．

[証明]　多項式 $X(z)$ を

$$X(z) = a\prod_{j=0}^{p}(z - \alpha_j)^{m_j}, \ a \neq 0, \ \sum_{j=0}^{p} m_j = n+1, \ \alpha_i \neq \alpha_j \ (i \neq j)$$

として，まず $f[X]$ の複素積分表示を示す．$\deg X = n+1$ であるので，式 (5.1) の右辺は高々 n 次の多項式である．これを $\pi_n(z)$ とおく．$\pi_n(z)$ が $f(z)$ の $X(z)$ の零点における補間，すなわち

$$\pi_n^{(k)}(\alpha_j) = f^{(k)}(\alpha_j), \ 0 \le k \le m_j - 1, \ 0 \le j \le p$$

を示せば，補間多項式の一意性より $\pi_n(z) = f[X]$ がいえるので，これを示す．ここで，$\pi_n^{(k)}(z)$，$f^{(k)}(z)$ は，それぞれ $\pi_n(z)$，$f(z)$ の k 階導関数を意味する．

$$\begin{aligned} \pi_n^{(k)}(\alpha_j) &= \frac{1}{2\pi i}\oint_C \frac{\partial^k}{\partial z^k}\left\{\left(1 - \frac{X(z)}{X(\zeta)}\right)\frac{f(\zeta)}{\zeta - z}\right\}\Big|_{z=\alpha_j} d\zeta \\ &= \frac{1}{2\pi i}\oint_C \sum_{l=0}^{k}\binom{k}{l}\frac{\partial^{k-l}}{\partial z^{k-l}}\left(1 - \frac{X(z)}{X(\zeta)}\right)\Big|_{z=\alpha_j}\frac{\partial^l}{\partial z^l}\left(\frac{1}{\zeta - z}\right)\Big|_{z=\alpha_j} f(\zeta)d\zeta \end{aligned}$$

ここで，

$$\frac{\partial^{k-l}}{\partial z^{k-l}}\left(1 - \frac{X(z)}{X(\zeta)}\right)\Big|_{z=\alpha_j} = \begin{cases} 1, & l = k \\ 0, & l \neq k \end{cases}$$

$$\left.\frac{\partial^l}{\partial z^l}\left(\frac{1}{\zeta - z}\right)\right|_{z=\alpha_j} = \frac{l!}{(\zeta - \alpha_j)^{l+1}}$$

より

$$\begin{aligned}\pi_n^{(k)}(\alpha_j) &= \frac{k!}{2\pi i}\oint_C \frac{f(\zeta)}{(\zeta - \alpha_j)^{k+1}}d\zeta \\ &= f^{(k)}(\alpha_j)\end{aligned}$$

となる．ゆえに，$\pi_n(z)$ は，$f(z)$ の $X(z)$ の零点における補間多項式となる．
$f[X;z]$ の複素積分表示は，$f(z)$，$f[X]$ の複素積分表示を使って得られる．■

5.3 差分商の評価

多項式補間において，差分商の評価は，補間の誤差評価に用いられる．

定理 5.3.1 $f(z)$ が閉円板 $\overline{D}_R$ で正則であり，$n+1$ 次多項式 $X(z)$ の零点がすべて $\overline{D}_R$ の内部にあるとき，$f(z)$ の $X(z)$ の零点上における差分商は，開円板 D_R 上で

$$|f[X;z]| \le \max_{|\zeta|=R}\left|\frac{[f(\zeta)]_{n+1}^{\infty}}{X(\zeta)}\right|\frac{1}{1-|z|/R} \tag{5.3}$$

と評価される．

[証明]　定理 5.2.1 の 式 (5.2) より $f(z)$ の $X(z)$ の零点上における差分商は

$$f[X;z] = \frac{1}{2\pi i}\int_{|\zeta|=R}\frac{f(\zeta)}{X(\zeta)(\zeta - z)}d\zeta. \tag{5.4}$$

他方

$$f(z) = [f(z)]_0^n + [f(z)]_{n+1}^{\infty}$$

と $[f(z)]_0^n$ の $X(z)$ の零点上における差分商は零であるので，式 (5.4) は

$$f[X;z] = \frac{1}{2\pi i}\int_{|\zeta|=R}\frac{[f(\zeta)]_{n+1}^{\infty}}{X(\zeta)(\zeta - z)}d\zeta. \tag{5.5}$$

この式を上から評価すれば

$$\begin{aligned}|f[X;z]| &\le \frac{1}{2\pi}\int_0^{2\pi}\left|\frac{[f(Re^{i\theta})]_{n+1}^{\infty}}{X(Re^{i\theta})}\right|\frac{R}{R-|z|}d\theta \\ &\le \max_{|\zeta|=R}\left|\frac{[f(\zeta)]_{n+1}^{\infty}}{X(\zeta)}\right|\frac{1}{1-|z|/R}\end{aligned}$$ ■

定理 5.3.2 [16] $f(z)$ が有理関数であって

$$\lim_{z\to\infty}\frac{f(z)}{X(z)} = 0 \tag{5.6}$$

を満たすとき，$f(z)$ の $X(z)$ の零点上における差分商は

$$f[X;z] = -\sum_j \frac{\mathrm{Res}f(\zeta_j)}{X(\zeta_j)(\zeta_j - z)} \tag{5.7}$$

と書ける．ここで，ζ_j は $f(z)$ の極で，$\mathrm{Res}f(\zeta_j)$ は ζ_j における $f(z)$ の留数である．

[証明]　差分商 $f[X;z]$ の複素積分表示 (5.2) において，積分路 C を図 5.1 のようにとる．ここで，R は十分大きくとる．このとき差分商は

$$f[X;z] = \frac{R}{2\pi}\int_0^{2\pi}\frac{f(Re^{i\theta})e^{i\theta}}{X(Re^{i\theta})(Re^{i\theta} - z)}d\theta - \sum_j\frac{\mathrm{Res}f(\zeta_j)}{X(\zeta_j)(\zeta_j - z)}$$

となる．ここで，上式右辺の第 1 項の絶対値は

$$\frac{R}{2\pi}\left|\int_0^{2\pi}\frac{f(Re^{i\theta})e^{i\theta}}{X(Re^{i\theta})(Re^{i\theta} - z)}d\theta\right| \le \max_{|\zeta|=R}\left|\frac{f(\zeta)}{X(\zeta)}\right|\frac{R}{R-|z|} \to 0\ (R\to\infty)$$

となり，したがって (5.7) が成り立つ．■

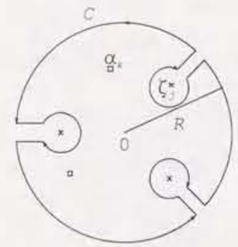

図 5.1: × は $f(z)$ の極，□ は $X(z)$ の零点

5.4 多項式補間の誤差評価

差分商の評価を用いて，多項式補間の誤差評価を行う．補間に用いる標本点集合 S およびそれを含むある閉円板 $\overline{D}_\rho$ において，n 次多項式 $X_n(z)$ は，その零点が全て S の要素であり，$\overline{D}_\rho$ を含む任意の閉円板 $\overline{D}_R$ 上で次の条件を満足するものとする．

任意の正数 $r < R$ に対して

$$0 < {}^{\exists}\delta < 1,\ {}^{\exists}M > 0 \text{ s.t.}$$

$$\left|\frac{X_n(z)}{X_n(\zeta)}\right| \leq M\delta^n \quad \text{for } |\zeta| = R,\ |z| \leq r. \tag{5.8}$$

条件 (5.8) を満たす代表的な例を挙げる.

例 5.4.1 $S = \{0\}$, $X_n(z) = z^n$ の場合.

S を含む任意の閉円板 $\overline{D}_\rho$ において, $\overline{D}_\rho$ を含む任意の閉円板 $\overline{D}_R$ と任意の正数 $r < R$ に対して

$$\delta = \frac{r}{R} \tag{5.9}$$

とおけば, $0 < \delta < 1$ で

$$\left|\frac{X_n(z)}{X_n(\zeta)}\right| = \left|\frac{z}{\zeta}\right|^n \leq \delta^n, \text{ for } |\zeta| = R,\ |z| \leq r$$

となり, 多項式 $X_n(z)$ は, 与えられた任意の閉円板 $\overline{D}_\rho$ に対して, それを含む任意の閉円板 $\overline{D}_R$ 上で, 条件 (5.8) を満たす.

例 5.4.2 $S = \{z \mid |z| = 1\}$, $X_n(z) = z^n - 1$ の場合.

S を含む任意の閉円板 $\overline{D}_\rho$ $(\rho > 1)$ において, $\overline{D}_\rho$ を含む任意の閉円板 $\overline{D}_R$ と任意の正数 $r < R$ に対して

$$\delta = \max\left(\frac{r}{R}, \frac{1}{R}\right) \tag{5.10}$$

とおけば, $0 < \delta < 1$ で

$$\begin{aligned}\left|\frac{X_n(z)}{X_n(\zeta)}\right| &= \left|\frac{z^n - 1}{\zeta^n - 1}\right| \\ &\leq \frac{|z|^n + 1}{R^n - 1} \\ &\leq \frac{(r/R)^n + (1/R)^n}{1 - (1/R)^n} \\ &\leq \frac{2}{1 - 1/R} \cdot \delta^n, \text{ for } |\zeta| = R,\ |z| \leq r\end{aligned}$$

となり, 多項式 $X_n(z)$ は, 単位円を含む任意の閉円板$\overline{D}_\rho$ $(\rho > 1)$ に対して, それを含む任意の閉円板 $\overline{D}_R$ 上で, 条件 (5.8) を満たす.

例 5.4.1 は Maclaurin 展開による多項式補間, Padé 展開による有理関数補間の誤差評価に用いられ, 例 5.4.2 は FFT による多項式補間, 単位円周上における有理関数補間の誤差評価に用いられる.

定理 5.4.1 定理 5.3.1の条件の下で, $n+1$ 次多項式 $X_{n+1}(z)$ は, その零点が全て Sの要素であり, 閉円板$\overline{D}_R$ $(\supset S)$ 上で条件 (5.8) を満足しているとする. このとき, $f(z)$ の $X_{n+1}(z)$ の零点上における補間の誤差を

$$\varepsilon_{n+1}(z) = f(z) - f[X_{n+1}]$$

とすれば, ある正数 M が存在して

$$|\varepsilon_{n+1}(z)| \leq \frac{\max\limits_{|\zeta|=R}\left|[f(\zeta)]_{n+1}^{\infty}\right|}{1 - r/R} \cdot M\delta^{n+1} \text{ for } |z| \leq r \tag{5.11}$$

が成り立つ. ここで, r は $r < R$ を満たす正の定数である.

[証明] 多項式 $X_{n+1}(z)$ は閉円板 $\overline{D}_R$ 上で条件 (5.8) を満たしているので, 正数 $r < R$ に対して, 正の数 $\delta < 1$ と M が存在して

$$\left|\frac{X_{n+1}(z)}{X_{n+1}(\zeta)}\right| \leq M\delta^{n+1} \quad \text{for } |\zeta| = R,\ |z| \leq r. \tag{5.12}$$

一方, 関数 $f(z)$ は $\overline{D}_R$ で正則であるので, 定理 5.3.1 より, $f(z)$ の $X_{n+1}(z)$ の零点上における差分商 $f[X_{n+1}; z]$ は, 開円板 D_R 上で

$$|f[X_{n+1}; z]| \leq \max_{|\zeta|=R}\left|\frac{[f(\zeta)]_{n+1}^{\infty}}{X_{n+1}(\zeta)}\right| \frac{1}{1 - |z|/R} \tag{5.13}$$

と評価される. この評価式と (5.12) を用いれば, 閉円板 $\overline{D}_r$ 上における誤差 $\varepsilon_{n+1}(z)$ は

$$\begin{aligned}|\varepsilon_{n+1}(z)| &= |f[X_{n+1}; z]X_{n+1}(z)| \\ &\leq \max_{|\zeta|=R}\left|\frac{[f(\zeta)]_{n+1}^{\infty}}{X_{n+1}(\zeta)}\right| \frac{|X_{n+1}(z)|}{1 - |z|/R} \\ &\leq \frac{\max\limits_{|\zeta|=R}\left|[f(\zeta)]_{n+1}^{\infty}\right|}{1 - r/R} \cdot M\delta^{n+1}\end{aligned}$$

となり, 定理が成り立つ. ■

この定理において, Maclaurin 展開および FFT による多項式補間の誤差は, 例 5.4.1, 5.4.2 を用いてそれぞれ

$$\begin{aligned}|\varepsilon_{n+1}(z)| &\leq \frac{\max\limits_{|\zeta|=R}\left|[f(\zeta)]_{n+1}^{\infty}\right|}{1 - r/R} \cdot (r/R)^{n+1} \text{ for } |z| \leq r, \\ |\varepsilon_{n+1}(z)| &\leq \frac{\max\limits_{|\zeta|=R}\left|[f(\zeta)]_{n+1}^{\infty}\right|}{1 - r/R} \cdot \frac{2}{1 - 1/R} \cdot (r/R)^{n+1} \text{ for } |z| \leq r\end{aligned}$$

と評価される.

5.5 有理関数補間の誤差評価

有理関数補間において，対象とする関数 $f(z)$ は単連結領域 D で有理型であるとし，標本点集合 S で正則であるとする．ただし，S は D の有界閉部分集合とする．多項式 $X(z)$ の零点が全て S の要素であるとき，次の条件を満たす有理式 $C(z)/A(z)$ を $f(z)$ の $X(z)$ の零点上における有理補間式と呼ぶ．

$$\begin{cases} Af - C = (Af)[X:z]X, \\ \deg A + \deg C < \deg X. \end{cases} \tag{5.14}$$

上式において，特に $S = \{0\}$ で $X(z) = z^n$ のとき，有理式 $C(z)/A(z)$ を $f(z)$ の Padé 近似式と呼ぶ．分母の次数を固定したときの Padé 近似の誤差表示は Montessus が行った [2]．本節では，これを一般化して，有理補間の誤差表示を行う．有理補間に関する性質をいくつか挙げる．

定理 5.5.1 (5.14) で定義された $f(z)$ の有理補間式において，$\deg Q \le \deg A$ を満たす任意の多項式 $Q(z)$ に対して

$$Af - C = \frac{1}{Q}(AQf)[X:z]X \tag{5.15}$$

が成立する．

[証明]　(5.14) 式の両辺に $Q(z)$ を掛ければ

$$QAf - QC = Q(Af)[X:z]X \tag{5.16}$$

となる．上式において，$\deg Q \le \deg A$ より

$$\begin{aligned} \deg QC &= \deg Q + \deg C \\ &\le \deg A + \deg C < \deg X \end{aligned}$$

であるので，両辺 $X(z)$ で mod をとれば

$$(QAf)[X] - QC = 0$$

すなわち

$$(QAf)[X] = QC \tag{5.17}$$

となる．一方 $Q(z)A(z)f(z)$ は $X(z)$ の零点近傍で解析的であるので補間と差分商を用いて

$$QAf = (QAf)[X] + (QAf)[X:z]X \tag{5.18}$$

と書ける．この式と (5.17) より (5.16) 式において右辺は

$$Q(Af)[X:z]X = (AQf)[X:z]X \tag{5.19}$$

となるので，このとき両辺を $Q(z)$ で割れば (5.15) が得られる．■

この定理において，零点が全て S に含まれる多項式 $X(z)$ が条件 (5.8) を満たすとき，定理 5.4.1 より次の誤差評価が得られる．

定理 5.5.2 関数 $f(z)$ は S を含む閉円板 $\overline{D}_R$ 上で有理型であり，$n+1$ 次多項式 $X_{n+1}(z)$ は，その零点が全て S に含まれ，$\overline{D}_R$ 上で条件 (5.8) を満足するものとする．このとき，$f(z)$ の $\overline{D}_R$ 上の 全ての極を零点に持つ多項式を $Q(z)$ として

$$g(z) = Q(z)f(z)$$

としたときに，$f(z)$ の $X_{n+1}(z)$ の零点上における有理補間式 $C^{[n]}(z)/A^{[n]}(z)$ に対して，その分母が

$$\deg A^{[n]} \ge \deg Q \tag{5.20}$$

を満たせば，正の数 M が存在して，閉円板 $\overline{D}_r$ $(r < R)$ 上で

$$\left|A^{[n]}(z)f(z) - C^{[n]}(z)\right| \le \frac{\max\limits_{|\zeta|=R}\left|[A^{[n]}(\zeta)g(\zeta)]_{n+1}^{\infty}\right|}{|Q(z)|(1-r/R)} \cdot M\delta^{n+1} \tag{5.21}$$

が成り立つ．

定理 5.5.2 において，$n+1$ 次多項式 $X_{n+1}(z)$ が特に z^{n+1} あるいは $z^{n+1}-1$ の場合，例 (5.4.1)，(5.4.2) より誤差評価式 (5.21) の右辺の定数 M は具体的にはそれぞれ 1，$2/(1-1/R)$ となり．δ はいずれも r/R となる．

次の定理は，関数 $f(z)$ の有理補間式の分母が，$f(z)$ の極の上で零に収束する速さを示す．

定理 5.5.3 定理 5.5.2 の条件の下で，閉円板 $\overline{D}_R$ における関数 $f(z)$ の 極は，ν 個の単純な極 $\{\zeta_j\}_{j=1}^{\nu}$ だけとする．このとき，$f(z)$ の $X_{n+1}(z)$ の零点上における有理補間式 $C^{[n]}(z)/A^{[n]}(z)$ の分母 $A^{[n]}(z)$ が $\deg A^{[n]} \ge \deg Q = \nu$ を満せば

$$\frac{\left\|A^{[n]}[Q]\right\|}{\max\limits_{|\zeta|=R}|A^{[n]}[Q;\zeta]|} = O(\delta^{n+1}) \ (n \to \infty) \tag{5.22}$$

が成立する．ここで，記号 $\|\cdot\|$ は，多項式の係数に関するノルムを表す．

[証明]　関数 $f(z)$ の $X_{n+1}(z)$ における有理補間式 $C^{[n]}(z)/A^{[n]}(z)$ と多項式 $Q(z)$ において，$\deg A^{[n]} \ge \deg Q$ のとき，定理 5.5.1 の (5.17) より $(A^{[n]}g)[X_{n+1}] = Q(z)C^{[n]}(z)$ が成り立つ．ここで，$g(z) = Q(z)f(z)$ である．これより

$$(A^{[n]}g)[X_{n+1}]]_{z=\zeta_j} = 0, \quad 1 \le j \le \nu. \tag{5.23}$$

上式左辺の複素積分表示は

$$(A^{[n]}g)[X_{n+1}]]_{z=\zeta_j} = \frac{1}{2\pi i}\int_{|\zeta|=R}\left(1 - \frac{X_{n+1}(\zeta_j)}{X_{n+1}(\zeta)}\right)\frac{A^{[n]}(\zeta)g(\zeta)}{\zeta - \zeta_j}d\zeta \tag{5.24}$$

となる．ここで

$$\begin{aligned} Q_0(z) &= 1, \\ Q_k(z) &= \prod_{j=1}^{k}(z-\zeta_j),\ (1 \le k < \nu) \end{aligned}$$

とおき

$$A^{[n]}[Q] = \sum_{k=1}^{\nu} r_k^{[n]} Q_{k-1}$$

として，$A^{[n]}(z) = A^{[n]}[Q] + A^{[n]}[Q;z]Q$ を式 (5.24) に代入すれば

$$\begin{aligned} (A^{[n]}g)[X_{n+1}]|_{z=\zeta_j} &= \sum_{k=1}^{\nu} \frac{1}{2\pi i}\int_{|\zeta|=R}\left(1-\frac{X_{n+1}(\zeta_j)}{X_{n+1}(\zeta)}\right)\frac{Q_{k-1}(\zeta)g(\zeta)}{\zeta-\zeta_j}d\zeta \cdot r_k^{[n]} \\ &\quad + \frac{1}{2\pi i}\int_{|\zeta|=R}\left(1-\frac{X_{n+1}(\zeta_j)}{X_{n+1}(\zeta)}\right)\frac{A^{[n]}[Q;\zeta]Q(\zeta)g(\zeta)}{\zeta-\zeta_j}d\zeta. \end{aligned} \tag{5.25}$$

(5.25) 式右辺において

$$\begin{aligned} c_{j,k}^{[n]} &= \frac{1}{2\pi i}\int_{|\zeta|=R}\left(1-\frac{X_{n+1}(\zeta_j)}{X_{n+1}(\zeta)}\right)\frac{Q_{k-1}(\zeta)g(\zeta)}{\zeta-\zeta_j}d\zeta \\ d_j^{[n]} &= -\frac{1}{2\pi i}\int_{|\zeta|=R}\left(1-\frac{X_{n+1}(\zeta_j)}{X_{n+1}(\zeta)}\right)\frac{A^{[n]}[Q;\zeta]Q(\zeta)g(\zeta)}{\zeta-\zeta_j}d\zeta \end{aligned}$$

とおけば，式 (5.23) は

$$\sum_{k=1}^{\nu} c_{j,k}^{[n]} r_k^{[n]} = d_j^{[n]}, \quad 1 \le j \le \nu \tag{5.26}$$

となる．$c_{j,k}^{[n]}$，$d_j^{[n]}$ は特に

$$c_{j,k}^{[n]} = \begin{cases} -\dfrac{1}{2\pi i}\displaystyle\int_{|\zeta|=R}\frac{X_{n+1}(\zeta_j)}{X_{n+1}(\zeta)}\frac{Q_{k-1}(\zeta_j)g(\zeta)}{\zeta-\zeta_j}d\zeta, & j<k \\ Q_{k-1}(\zeta_j)g(\zeta_j) - \dfrac{1}{2\pi i}\displaystyle\int_{|\zeta|=R}\frac{X_{n+1}(\zeta_j)}{X_{n+1}(\zeta)}\frac{Q_{k-1}(\zeta)g(\zeta)}{\zeta-\zeta_j}d\zeta, & j \ge k \end{cases} \tag{5.27}$$

$$d_j^{[n]} = \frac{1}{2\pi i}\int_{|\zeta|=R}\frac{X_{n+1}(\zeta_j)}{X_{n+1}(\zeta)}\frac{A^{[n]}[Q;\zeta]Q(\zeta)g(\zeta)}{\zeta-\zeta_j}d\zeta \tag{5.28}$$

となる．

上式において，$\left|\frac{X_{n+1}(\zeta_j)}{X_{n+1}(\zeta)}\right|$ は $\overline{D}_R$ 上で条件 (5.8) を満たすので，この閉円板と各番号 j に対して

$$0 < {}^{\exists}\delta_j < 1 \text{ s.t.} \left|\frac{X_{n+1}(\zeta_j)}{X_{n+1}(\zeta)}\right| = O(\delta_j^{n+1})\ (n\to\infty) \quad \text{for } |\zeta| = R$$

となる．ここで $\delta = \max_{1\le j\le\nu}\delta_j$ とおけば

$$\left|\frac{X_{n+1}(\zeta_j)}{X_{n+1}(\zeta)}\right| = O(\delta^{n+1})\ (n\to\infty) \quad \text{for } |\zeta| = R,\ 1 \le j \le \nu.$$

この評価式より，式 (5.27)，(5.28) はそれぞれ

$$c_{j,k}^{[n]} = \begin{cases} O(\delta^{n+1}), & j<k \\ Q_{k-1}(\zeta_j)g(\zeta_j) + O(\delta^{n+1}), & j \ge k \end{cases} \tag{5.29}$$

$$|d_j^{[n]}| = \max_{|\zeta|=R}|A^{[n]}[Q;\zeta]| \cdot O(\delta^{n+1}) \tag{5.30}$$

となる．したがって (5.26) 式は

$$(L+\delta L)\boldsymbol{r}^{[n]} = \boldsymbol{d}^{[n]} \tag{5.31}$$

と書ける．ここで，上式の係数行列およびベクトルは，それぞれ

$$L = \begin{pmatrix} g(\zeta_1) & & & 0 \\ g(\zeta_2) & Q_1(\zeta_2)g(\zeta_2) & & \\ \vdots & \vdots & \ddots & \\ g(\zeta_\nu) & Q_1(\zeta_\nu)g(\zeta_\nu) & \dots & Q_{\nu-1}(\zeta_\nu)g(\zeta_\nu) \end{pmatrix}, \quad \|\delta L\| = O(\delta^{n+1})$$

$$\boldsymbol{r}^{[n]} = (r_1^{[n]}, r_2^{[n]}, \dots, r_\nu^{[n]})^T, \quad \boldsymbol{d}^{[n]} = (d_1^{[n]}, d_2^{[n]}, \dots, d_\nu^{[n]})^T$$

である．行列 L は下三角行列であり，その対角要素は零でないので逆をもつ．また係数行列である $L+\delta L$ について，$\|L^{-1}\delta L\| < 1$ となるように n を十分大きくとれば，$L+\delta L$ も逆をもつ．このとき，線形方程式 (5.31) は，$\boldsymbol{r}^{[n]}$ について解くことができる．

$$\boldsymbol{r}^{[n]} = (L+\delta L)^{-1}\boldsymbol{d}^{[n]}.$$

上式をノルムで評価すれば

$$\begin{aligned} \left\|\boldsymbol{r}^{[n]}\right\| &\le \|L^{-1}\|\|(I+L^{-1}\delta L)^{-1}\|\left\|\boldsymbol{d}^{[n]}\right\| \\ &\le \frac{\|L^{-1}\|}{1-\|L^{-1}\delta L\|}\left\|\boldsymbol{d}^{[n]}\right\|. \end{aligned}$$

$\left\|\boldsymbol{d}^{[n]}\right\| = \max_{|\zeta|=R}|A^{[n]}[Q;\zeta]| \cdot O(\delta^{n+1})$ であるので，$\left\|\boldsymbol{r}^{[n]}\right\| = \max_{|\zeta|=R}|A^{[n]}[Q;\zeta]| \cdot O(\delta^{n+1})$ となる．他方

$$\left\|A^{[n]}[Q]\right\| \le \sum_{k=1}^{\nu}\left|r_k^{[n]}\right|\|Q_{k-1}\|$$

である．よって，$\left\|A^{[n]}[Q]\right\| = \max_{|\zeta|=R}|A^{[n]}[Q;\zeta]| \cdot O(\delta^{n+1})$ となり，定理が示された．■

この定理の下三角行列 L の上から 2 番目以降の対角要素

$$Q_1(\zeta_2)g(\zeta_2),\ Q_2(\zeta_3)g(\zeta_3),\ \cdots,\ Q_{\nu-1}(\zeta_\nu)g(\zeta_\nu)$$

において，$f(z)$ の極 $\{\zeta_j\}_{j=1}^{\nu}$ が互いに近接しているとき，上の対角要素もそれに伴って小さくなるので，L^{-1} のノルムも大きくなる．これは極が近接していない場合に比べて，有理関数補間による極の近似の精度が悪くなることを意味する．

定理 5.5.4 定理 5.5.3 において，関数 $f(z)$ が 閉円板 $\overline{D}_R$ で $\zeta_j\ (1 \le j \le q)$ を μ_j 位の極に持ち，それ以外の極はないとする．このとき，$f(z)$ の $X_{n+1}(z)$ の零点上における有理補間式 $C^{[n]}(z)/A^{[n]}(z)$ の分母 $A^{[n]}(z)$ が $\deg A^{[n]} \ge \deg Q = \sum_{j=1}^{q}\mu_j$ を満せば

$$\frac{\left\|A^{[n]}[Q]\right\|}{\max\limits_{|\zeta|=R}|A^{[n]}[Q;\zeta]|}\cdot = O(n^{\mu_{\max}-1}\delta^{n+1})\ (n\to\infty) \tag{5.32}$$

が成立する．ここで，$\mu_{\max} = \max\limits_{1\le j\le q}\mu_j$ である．

[証明]　$(A^{[n]}g)[X_{n+1}] = Q(z)C^{[n]}(z)$ より

$$\frac{d^k}{dz^k}\left((A^{[n]}g)[X_{n+1}]\right)\Big|_{z=\zeta_i} = 0,\ 0\le k\le \mu_i-1,\ 1\le i\le q. \tag{5.33}$$

上式右辺の複素積分表示は

$$\begin{aligned}\frac{d^k}{dz^k}\left((A^{[n]}g)[X_{n+1}]\right)\Big|_{z=\zeta_i} &= \frac{k!}{2\pi i}\int_{|\zeta|=R}\left(1-\frac{X_{n+1}(\zeta_i)}{X_{n+1}(\zeta)}\right)\frac{A^{[n]}(\zeta)g(\zeta)}{(\zeta-\zeta_i)^{k+1}}d\zeta\\ &\quad -\sum_{\mu=1}^{k}\binom{k}{\mu}\frac{(k-\mu)!}{2\pi i}\int_{|\zeta|=R}\frac{X_{n+1}^{(\mu)}(\zeta_i)}{X_{n+1}(\zeta)}\frac{A^{[n]}(\zeta)g(\zeta)}{(\zeta-\zeta_i)^{k-\mu+1}}d\zeta\end{aligned} \tag{5.34}$$

となる．ただし，$X_{n+1}^{(k-\mu)}(\zeta_i)$ は，$X_{n+1}(z)$ の ζ_i における $k-\mu$ 階導関数値を表す．ここで

$$\begin{aligned}Q_0(z) &= 1,\\ Q_{s_{j-1}+l}(z) &= \prod_{k=1}^{j-1}(z-\zeta_k)^{\mu_k}\times(z-\zeta_j)^l,\ 0\le l<\mu_j,\ 1\le j\le q\\ &\quad \text{ただし，}s_0=0,\ s_i=\sum_{k=1}^{i}\mu_k\end{aligned}$$

とおき

$$A^{[n]}[Q] = \sum_{j=1}^{q}\sum_{l=1}^{\mu_j} r_{s_{j-1}+l}^{[n]}Q_{s_{j-1}+l-1} \tag{5.35}$$

として，$A^{[n]}(z) = A^{[n]}[Q] + A^{[n]}[Q;z]Q$ を式 (5.34) に代入すれば

$$\begin{aligned}&\frac{d^k}{dz^k}\left((A^{[n]}g)[X_{n+1}]\right)\Big|_{z=\zeta_i}\\ &= \sum_{j=1}^{q}\sum_{l=1}^{\mu_j}\left\{\frac{k!}{2\pi i}\int_{|\zeta|=R}\left(1-\frac{X_{n+1}(\zeta_i)}{X_{n+1}(\zeta)}\right)\frac{Q_{s_{j-1}+l-1}(\zeta)g(\zeta)}{(\zeta-\zeta_i)^{k+1}}d\zeta\right.\\ &\quad \left.-\sum_{\mu=1}^{k}\binom{k}{\mu}\frac{(k-\mu)!}{2\pi i}\int_{|\zeta|=R}\frac{X_{n+1}^{(\mu)}(\zeta_i)}{X_{n+1}(\zeta)}\frac{Q_{s_{j-1}+l-1}(\zeta)g(\zeta)}{(\zeta-\zeta_i)^{k-\mu+1}}d\zeta\right\}r_{s_{j-1}+l}^{[n]}\\ &\quad +\frac{k!}{2\pi i}\int_{|\zeta|=R}\left(1-\frac{X_{n+1}(\zeta_i)}{X_{n+1}(\zeta)}\right)\frac{A^{[n]}[Q;\zeta]Q(\zeta)g(\zeta)}{(\zeta-\zeta_i)^{k+1}}d\zeta\\ &\quad -\sum_{\mu=1}^{k}\binom{k}{\mu}\frac{(k-\mu)!}{2\pi i}\int_{|\zeta|=R}\frac{X_{n+1}^{(\mu)}(\zeta_i)}{X_{n+1}(\zeta)}\frac{A^{[n]}[Q;\zeta]Q(\zeta)g(\zeta)}{(\zeta-\zeta_i)^{k-\mu+1}}d\zeta\end{aligned}$$

上式右辺において

$$\begin{aligned}c_{s_{i-1}+k,s_{j-1}+l}^{[n]} &= \frac{(k-1)!}{2\pi i}\int_{|\zeta|=R}\left(1-\frac{X_{n+1}(\zeta_i)}{X_{n+1}(\zeta)}\right)\frac{Q_{s_{j-1}+l-1}(\zeta)g(\zeta)}{(\zeta-\zeta_i)^{k}}d\zeta\\ &\quad -\sum_{\mu=1}^{k-1}\binom{k-1}{\mu}\frac{(k-\mu-1)!}{2\pi i}\int_{|\zeta|=R}\frac{X_{n+1}^{(\mu)}(\zeta_i)}{X_{n+1}(\zeta)}\frac{Q_{s_{j-1}+l-1}(\zeta)g(\zeta)}{(\zeta-\zeta_i)^{k-\mu}}d\zeta\\ d_{s_{i-1}+k}^{[n]} &= -\frac{(k-1)!}{2\pi i}\int_{|\zeta|=R}\left(1-\frac{X_{n+1}(\zeta_i)}{X_{n+1}(\zeta)}\right)\frac{A^{[n]}[Q;\zeta]Q(\zeta)g(\zeta)}{(\zeta-\zeta_i)^{k}}d\zeta\\ &\quad +\sum_{\mu=1}^{k-1}\binom{k-1}{\mu}\frac{(k-\mu-1)!}{2\pi i}\int_{|\zeta|=R}\frac{X_{n+1}^{(\mu)}(\zeta_i)}{X_{n+1}(\zeta)}\frac{A^{[n]}[Q;\zeta]Q(\zeta)g(\zeta)}{(\zeta-\zeta_i)^{k-\mu}}d\zeta\end{aligned}$$

とおけば，式 (5.33) は

$$\sum_{j=1}^{q}\sum_{l=1}^{\mu_j}c_{s_{i-1}+k,s_{j-1}+l}^{[n]}\cdot r_{s_{j-1}+l}^{[n]} = d_{s_{i-1}+k}^{[n]},\ 1\le k\le\mu_i,\ 1\le i\le q \tag{5.36}$$

となり，$c_{s_{i-1}+k,s_{j-1}+l}^{[n]}$，$d_{s_{i-1}+k}^{[n]}$ は特に

$$\begin{aligned}c_{s_{i-1}+k,s_{j-1}+l}^{[n]} &= \begin{cases}-\sum\limits_{\mu=0}^{k-1}\dfrac{1}{\mu!}\dfrac{(k-1)!}{2\pi i}\displaystyle\int_{|\zeta|=R}\frac{X_{n+1}^{(\mu)}(\zeta_i)}{X_{n+1}(\zeta)}\frac{Q_{s_{j-1}+l-1}(\zeta)g(\zeta)}{(\zeta-\zeta_i)^{k-\mu}}d\zeta\\ \qquad\text{for } i<j \text{ or } i=j \text{ and } k<l\\ \dfrac{(k-1)!}{2\pi i}\displaystyle\int_{|\zeta|=R}\frac{Q_{s_{j-1}+l-1}(\zeta)g(\zeta)}{(\zeta-\zeta_i)^{k}}d\zeta\\ \quad -\sum\limits_{\mu=0}^{k-1}\dfrac{1}{\mu!}\dfrac{(k-1)!}{2\pi i}\displaystyle\int_{|\zeta|=R}\frac{X_{n+1}^{(\mu)}(\zeta_i)}{X_{n+1}(\zeta)}\frac{Q_{s_{j-1}+l-1}(\zeta)g(\zeta)}{(\zeta-\zeta_i)^{k-\mu}}d\zeta\\ \qquad\text{for } i>j \text{ or } i=j \text{ and } k\ge l\end{cases}\\ d_{s_{i-1}+k}^{[n]} &= \sum_{\mu=0}^{k-1}\frac{1}{\mu!}\frac{(k-1)!}{2\pi i}\int_{|\zeta|=R}\frac{X_{n+1}^{(\mu)}(\zeta_i)}{X_{n+1}(\zeta)}\frac{A^{[n]}[Q;\zeta]Q(\zeta)g(\zeta)}{(\zeta-\zeta_i)^{k-\mu}}d\zeta\end{aligned}$$

となる．ここで，半径 R の円上で $\left|\frac{X_{n+1}^{(\mu)}(\zeta_i)}{X_{n+1}(\zeta)}\right|$ の上からの評価を行う．

$$X_{n+1}(z) = a\prod_{j=0}^{p}(z-\alpha_j)^{m_j},\ a\neq 0,\ \sum_{j=0}^{p} m_j = n+1,\ \alpha_i \neq \alpha_j\ (i\neq j)$$

としたときに，$X_{n+1}(z)$ の 1 階導関数は

$$\begin{aligned} X'_{n+1}(z) &= a\cdot\sum_{j=0}^{p} m_j(z-\alpha_j)^{m_j-1}\times\prod_{l\neq j}(z-\alpha_l)^{m_l} \\ &= \left(\sum_{j=0}^{p}\frac{m_j}{z-\alpha_j}\right)X_{n+1}(z) \end{aligned}$$

となる．これを用いれば，k 階導関数は

$$\begin{aligned} X_{n+1}^{(k)}(z) &= \frac{d^{k-1}}{dz^{k-1}}X'_{n+1}(z) \\ &= \frac{d^{k-1}}{dz^{k-1}}\left\{\left(\sum_{\mu=0}^{p}\frac{m_\mu}{z-\alpha_\mu}\right)X_{n+1}(z)\right\} \\ &= (k-1)!\cdot\sum_{l=0}^{k-1}\frac{(-1)^{k-l-1}}{l!}\left(\sum_{\mu=0}^{p}\frac{m_\mu}{(z-\alpha_\mu)^{k-l}}\right)X_{n+1}^{(l)}(z) \end{aligned}$$

となる．したがって

$$\begin{aligned} \left|\frac{X_{n+1}^{(k)}(\zeta_i)}{X_{n+1}(\zeta)}\right| &\leq (k-1)!\cdot\sum_{l=0}^{k-1}\frac{1}{l!}\left(\sum_{\mu=0}^{p}\frac{m_\mu}{|\zeta_i-\alpha_\mu|^{k-l}}\right)\left|\frac{X_{n+1}^{(l)}(\zeta_i)}{X_{n+1}(\zeta)}\right| \\ &\leq (k-1)!\cdot\frac{n+1}{\eta^k}\cdot\sum_{l=0}^{k-1}\frac{\eta^l}{l!}\left|\frac{X_{n+1}^{(l)}(\zeta_i)}{X_{n+1}(\zeta)}\right|. \end{aligned} \tag{5.37}$$

上式の評価において，$\eta = \min_{1\leq i\leq q}\min_{\alpha\in S}|\zeta_i-\alpha|$ とおき，$\sum_{\mu=0}^{p} m_\mu = n+1$ を用いた．上式を使って

$$\left|\frac{X_{n+1}^{(k)}(\zeta_i)}{X_{n+1}(\zeta)}\right| = O(n^k\delta^{n+1}) \quad \text{for } |\zeta| = R \tag{5.38}$$

が成立することを k に関する帰納法で示す．

$k=0$ のとき，多項式 $X_{n+1}(z)$ は $\overline{D}_R$ 上で条件 (5.8) を満足するので

$$\left|\frac{X_{n+1}(\zeta_i)}{X_{n+1}(\zeta)}\right| = O(\delta^{n+1}) \quad \text{for } |\zeta| = R$$

が成立する．

帰納法の仮定として，番号 l について $0\leq l<k$ まで

$$\left|\frac{X_{n+1}^{(l)}(\zeta_i)}{X_{n+1}(\zeta)}\right| \leq Mn^l\delta^{n+1}, \quad \text{for } |\zeta| = R$$

が成り立っているとする．ここで，Mは n に依存しない正の数である．このとき，式(5.37)より

$$\begin{aligned} \left|\frac{X_{n+1}^{(k)}(\zeta_i)}{X_{n+1}(\zeta)}\right| &\leq (k-1)!\cdot\frac{n+1}{\eta^k}\left(\sum_{l=0}^{k-1}\frac{\eta^l}{l!}n^l\right)M\delta^{n+1} \\ &< \frac{2(k-1)!}{\eta^k}M\left(\sum_{l=0}^{k-1}\frac{\eta^l}{l!}\right)\cdot n^k\delta^{n+1} \end{aligned}$$

ここで，右辺の $n^k\delta^{n+1}$ に掛かる係数は，n に依存しないので

$$\left|\frac{X_{n+1}^{(k)}(\zeta_i)}{X_{n+1}(\zeta)}\right| = O(n^k\delta^{n+1})$$

となり，$0\leq k<\mu_i$ について，(5.38) 式が成立する．

評価式 (5.38) より，$c_{s_{i-1}+k,s_{j-1}+l}^{[n]}$，$d_{s_{i-1}+k}^{[n]}$ は

$$c_{s_{i-1}+k,s_{j-1}+l}^{[n]} = \begin{cases} c_{s_{i-1}+k,s_{j-1}+l} + O(n^{k-1}\delta^{n+1}) \\ \qquad \text{for } i<j \text{ or } i=j \text{ and } k<l \\ O(n^{k-1}\delta^{n+1}) \\ \qquad \text{for } i>j \text{ or } i=j \text{ and } k\geq l \end{cases} \tag{5.39}$$

$$|d_{s_{i-1}+k}^{[n]}| = \max_{|\zeta|=R}|A^{[n]}[Q;\zeta]|\cdot O(n^{k-1}\delta^{n+1}) \tag{5.40}$$

ここで

$$c_{s_{i-1}+k,s_{j-1}+l} = \frac{(k-1)!}{2\pi i}\int_{|\zeta|=R}\frac{Q_{s_{j-1}+l-1}(\zeta)g(\zeta)}{(\zeta-\zeta_i)^k}d\zeta \tag{5.41}$$

とおいた．上式において，特に $i=j$，$k=l$ のときは

$$c_{s_{i-1}+k,s_{i-1}+k} = (k-1)!\cdot Q_{s_{i-1}}(\zeta_i)g(\zeta_i) \tag{5.42}$$

となり，このときの要素は零でない．

式 (5.36) を定理 5.5.3 と同様にして線形方程式で記述すれば

$$(L+\delta L)\boldsymbol{r}^{[n]} = \boldsymbol{d}^{[n]}$$

となる．ここで，L は下三角行列であり，その対角要素は式 (5.42) で与えられるので零でない．したがって L は逆をもつ．また δL，$\boldsymbol{d}^{[n]}$ は，それぞれ

$$\|\delta L\| = O(n^{\mu_{\max}-1}\delta^{n+1}),\ \left\|\boldsymbol{d}^{[n]}\right\| = \max_{|\zeta|=R}|A^{[n]}[Q;\zeta]|\cdot O(n^{\mu_{\max}-1}\delta^{n+1})$$

と評価される．ここで，$\mu_{\max} = \max_{1\le j\le q} \mu_j$ である．以下，定理 5.5.3 と同様の評価より

$$\frac{\left\|A^{[n]}[Q]\right\|}{\max_{|\zeta|=R}|A^{[n]}[Q;\zeta]|} = O(n^{\mu_{\max}-1}\delta^{n+1})\ (n\to\infty)$$

を得る．■

定理 5.5.4 は，定理 5.5.3 を一般化したものになっている．

定理 5.5.3，5.5.4 において，$n+1$ 次多項式 $X_{n+1}(z)$ が特に z^{n+1} あるいは $z^{n+1}-1$ のとき，零への収束の速さを支配する δ は

$$\delta = \frac{\max_j |\zeta_j|}{R} \tag{5.43}$$

となる．特に，$\deg A^{[n]} = \deg Q$ として，$A^{[n]}(z)$，$Q(z)$ の主係数をともに 1 にしたとき，$A^{[n]}(z)$ は $Q(z)$ に収束する．このとき，定理 5.5.2は，Montessus の定理を拡張した有理補間の収束定理になっている．

5.6 有理補間式の計算法

有理補間式の計算は，関数値あるいは関数のべき級数から計算されるが，ここでは後者の場合の計算法について考える．関数 $f(z)$ は，原点を中心としたある閉円板 D で有理型であり，標本点集合 S で正則であるとする．ここで，S は D の有界閉部分集合とする．2 つの多項式 $X(z)$，$Y(z)$ はそれぞれ零点が S の要素であり，$\deg X = n+1$，$\deg Y = \nu \ge 1$ として，$Y(z)$ の次数 ν を固定する．関数 $f(z)$ および多項式 $X(z)Y(z)$ に対して，z に関する $n+\nu$ 次の多項式 $F = f[XY]$ が与えられているとする．多項式 $X(z)$ を G として，F と G に拡張算法を適用して得られる拡張剰余列要素 (P,A,B) において，記号上 P を C で置き換え，拡張剰余列要素を (A,B,C) と並べ換えて表示することにする．この多項式 3 つ組は

$$AF + BG = C,\ \deg A + \deg C \le n \tag{5.44}$$

を満足している．このとき，有理式 $C(z)/A(z)$ は $f(z)$ の $X(z)$ の零点上における有理補間式であることが次の定理で示される．

定理 5.6.1 関数 $f(z)$ は，2 つの多項式 $X(z)$，$Y(z)$ の零点の近傍で解析的とし，$\deg X = n+1$，$\deg Y \ge 1$ とする．$F = f[XY]$，$G = X$ に対して，(5.44) を満たす多項式の 3 つ組 (A,B,C) において，有理式 $C(z)/A(z)$ は $f(z)$ の $X(z)$ の零点上における有理補間式であり

$$\begin{aligned} Af - C &= (Af)[X;z]X && (5.45)\\ &= -B[Y]X + (Af)[XY;z]XY && (5.46)\end{aligned}$$

が成り立つ．さらに，(5.46) 式の $-B[Y]$，$(Af)[XY;z]$ について

$$\begin{aligned} -B[Y] &= \left(\left(\frac{1}{Q}\right)[Y]\cdot(AQf)[X;z]\right)[Y], && (5.47)\\ (Af)[XY;z] &= \left(\left(\frac{1}{Q}\right)[Y]\cdot(AQf)[X;z]\right)[Y;z] + \left(\frac{1}{Q}\right)[Y;z]\cdot(AQf)[X;z] && (5.48)\end{aligned}$$

が成り立つ．ここで，$Q(z)$ は，$\deg Q \le \deg A$ を満たす任意の多項式である．

[証明]　関数 $f(z)$ を $X(z)Y(z)$ の零点上における補間 $f[XY] = F$ と差分商 $f[XY;z]$ で記述すれば

$$f(z) = F + f[XY;z]XY$$

となる．この式と (5.44) より

$$\begin{aligned} Af - C &= A(F + f[XY;z]XY) - C\\ &= (AF - C) + Af[XY;z]XY\\ &= -BX + Af[XY;z]XY = (-B + Af[XY;z]Y)X \end{aligned} \tag{5.49}$$

となる．上式最後の式において，$\deg C < \deg X$ でありかつ

$$Af = (Af)[X] + (Af)[X;z]X,\ \deg(Af)[X] < \deg X$$

と表されるので，$A(z)f(z)$ の $X(z)$ の零点上における差分商の一意性より

$$-B + Af[XY;z]Y = (Af)[X;z]$$

となり，(5.45) が示された．これと (5.44) とを併せれば，$C(z)/A(z)$ は $f(z)$ の $X(z)$ の零点上における有理補間式となる．(5.49) はさらに

$$\begin{aligned} (-B + Af[XY;z]Y)X &= -(B[Y] + B[Y;z]Y)X + Af[XY;z]XY\\ &= -B[Y]X + (-B[Y;z] + Af[XY;z])XY \end{aligned} \tag{5.50}$$

となり

$$\begin{aligned} \deg B[Y]X &= \deg B[Y] + \deg X\\ &< \deg Y + \deg X = \deg XY \end{aligned}$$

であるので，$A(z)f(z)$ の $X(z)Y(z)$ の零点上における差分商の一意性より

$$-B[Y;z] + Af[XY;z] = (Af)[XY;z]$$

がいえる．よって，(5.46) が得られる．

次に，$\deg Q \le \deg A$ である多項式 $Q(z)$ に対して，(5.47)，(5.48) を示す．定理 5.5.1 の結果より

$$\begin{aligned} Af - C &= \frac{1}{Q}(AQf)[X;z]X \\ &= \left\{\left(\frac{1}{Q}\right)[Y] + \left(\frac{1}{Q}\right)[Y;z]Y\right\} \cdot (AQf)[X;z]X \\ &= \left(\frac{1}{Q}\right)[Y] \cdot (AQf)[X;z]X + \left(\frac{1}{Q}\right)[Y;z] \cdot (AQf)[X;z]XY \\ &= \left(\left(\frac{1}{Q}\right)[Y] \cdot (AQf)[X;z]\right)[Y]X \\ &\quad + \left\{\left(\left(\frac{1}{Q}\right)[Y] \cdot (AQf)[X;z]\right)[Y;z] + \left(\frac{1}{Q}\right)[Y;z] \cdot (AQf)[X;z]\right\}XY \end{aligned}$$

となる．上式の最後の等式において，$A(z)f(z)$ の $X(z)Y(z)$ の零点における補間多項式とその差分商の一意性を用いれば，(5.47)，(5.48) が得られる．■

この定理の (5.47) より，有理補間誤差

$$Af - C = \frac{1}{Q}(AQf)[X;z]X,\ \deg A \ge \deg Q$$

の右辺の $(1/Q)\cdot(AQf)[X;z]$ を z の関数とみたときに，$-B[Y]$ はこの関数を $Y(z)$ の零点上で補間した多項式であることが分かる．$(Af)[XY;z]Y$ はそのときの補間誤差である．したがって，$-B[Y]X$ は有理補間誤差を $Y(z)$ の零点上で近似したものであるので，これを評価することにより有理関数補間の事後誤差評価ができる．実際の計算では，$B[Y]$ の評価を行う．$B[Y]=0$ ならば，(5.46) より $A(z)/C(z)$ は $f(z)$ の $X(z)Y(z)$ の零点上における補間多項式となる．(5.47)，(5.48) において，Y の次数 ν を固定したときに，差分商 $(AQf)[X;z]$ が

$$(AQf)[X;z] \to 0\ (n \to \infty)$$

ならば，$B[Y]$，$(Af)[XY;z]$ もそれにともなって零に収束する．多項式 $X(z)$ を具体的に与えて，差分商 $(AQf)[X;z]$ が零に収束する場合を考える．この差分商の評価については，定理 5.3.1 より

$$|(AQf)[X;z]| \le \max_{|\zeta|=R}\left|\frac{[(AQf)(\zeta)]_{n+1}^{\infty}}{X(\zeta)}\right| \frac{1}{1-|z|/R},\ \text{for } |z|<R$$

となる．

例 5.6.1 $X_{n+1}(z)=z^{n+1}$ の場合．

$$|(AQf)[X_{n+1};z]| \le \max_{|\zeta|=R}\frac{\left|[(AQf)(\zeta)]_{n+1}^{\infty}\right|}{R^{n+1}}\frac{1}{1-|z|/R},\ \text{for } |z|<R. \qquad (5.51)$$

例 5.6.2 $X_{n+1}(z)=z^{n+1}-1$ の場合．$R>1$，$R^{n+1}\gg 1$ ならば

$$\begin{aligned} |(AQf)[X_{n+1};z]| &\le \max_{|\zeta|=R}\frac{\left|[(AQf)(\zeta)]_{n+1}^{\infty}\right|}{R^{n+1}-1}\frac{1}{1-|z|/R} \\ &\approx \max_{|\zeta|=R}\frac{\left|[(AQf)(\zeta)]_{n+1}^{\infty}\right|}{R^{n+1}}\frac{1}{1-|z|/R},\ \text{for } |z|<R. \qquad (5.52) \end{aligned}$$

この 2 つの例において，正の定数 R が 1 より大きいとき，差分商 $(AQf)[X;z]$ はいずれも零に収束する．$(AQf)(z)$ の極が単位円周上から外に離れていれば，R は大きくとることができ，差分商の零への収束は速くなる．特に関数 $f(z)$ が有理関数の場合，差分商 $(AQf)[X;z]$ が恒等的に零になる有理補間式が存在する．次の定理でそれを示す．

定理 5.6.2 定理 5.6.1 の条件の下で，特に関数 $f(z)$ が

$$f(z)=P(z)/Q(z),\ \gcd(P,Q)=1,\ \deg P+\deg Q\le n$$

である有理関数で，その $X(z)$ の零点上での有理補間式を

$$C(z)/A(z),\ \deg A\ge\deg Q$$

としたときに，次の 2 条件は同値である．

$$\deg A=\deg Q. \qquad (5.53)$$

$$(AQf)[X;z]=0,\quad \gcd(A,C)=1. \qquad (5.54)$$

[証明] まず，(5.53) $\Longrightarrow$ (5.54) を示す．$AQf=AP$ および $\deg A=\deg Q$ より

$$\begin{aligned} \deg AQf &= \deg AP \\ &= \deg A+\deg P \\ &= \deg Q+\deg P<\deg X \end{aligned}$$

となるので，差分商 $(AQf)[X;z]$ は零となる．このとき，定理 5.5.1 より $Af-C=0$ すなわち $A=QC/P$ となる．ここで，$\gcd(P,Q)=1$ および $\deg A=\deg Q$ より，A はある零でない定数 λ によって，$A=\lambda Q$ と書ける．一方 C はこれを用いて，$C=AP/Q=\lambda P$ と書ける．したがって，$\gcd(A,C)=1$ が成立する．

次に，(5.54) $\Longrightarrow$ (5.53) を示す．$(AQf)[X;z]=0$ であるので，再び定理 5.5.1 より $Af-C=0$ すなわち $A=QC/P$ となる．ここで，$\gcd(P,C)=1$ であるので，A はある多項式 D によって $A=DQ$ と書ける．一方 C はこれを用いて，$C=AP/Q=DP$ と書ける．このとき

$$\begin{aligned} \gcd(A,C) &= \gcd(DP,DQ) \\ &= D\cdot\gcd(P,Q)=D \end{aligned}$$

となり，$\gcd(A,C)=1$ であるので，D は定数となる．よって，$\deg A=\deg Q$ となる．■

系 5.6.1 定理 5.6.2 において，条件 (5.53)，(5.54) のいずれか一方が成立すれば，多項式 A，C はある零でない定数 λ によってそれぞれ

$$A = \lambda Q,\ C = \lambda P$$

と表現され，$f(z) = C(z)/A(z)$ が成立する．

前述の定理において，有理関数 $f(z)$ の分母分子が未知であり，これらを有理補間により求める場合，条件 (5.53)，(5.54) のいずれか一方の成立を確かめる必要がある．$\gcd(A, C) = 1$ を仮定すれば，$(AQf)[X; z] = 0$ となることを調べればよい．これは $-B[Y]$ で推定することができる．なぜならば，$-B[Y]$ は，前に述べたように $1/Q \cdot (AQf)[X; z]$ の $Y(z)$ の零点上における補間多項式であるからである．

定理 5.6.3 定理 5.6.1 の多項式の 3 組 (A, B, C) において，$\gcd(A, C) = 1$ であるための必要十分条件は $\gcd(A, G) = 1$ である．すなわち，標本点である $X(z)$ の零点上に $A(z)$ の零点がないことである．

[証明]

$$\begin{aligned}\gcd(A, C) &= \gcd(A, AF + BG)\\ &= \gcd(A, BG)\\ &= \gcd(A, G)\end{aligned}$$

ここで，最後の等式は $\gcd(A, B) = 1$ による．これは p.44 の系 4.1.1において，剰余列要素 P が最低次数多項式であることから直ちに分かる．したがって，定理が成り立つ．■

この定理において，特に $G = z^{n+1}$ の場合，$\gcd(A, C) = 1$ であるための必要十分条件は，$A(z)$ の定数項が零でないことである．したがって，Padé 近似式の分母分子の既約性は，分母の定数項によって決まる．さらに $Y = z^{\nu}$ として，$B[Y] = 0$ のとき，$\gcd(A, C) = 1$ から $A(z)$ の定数項は零でない．したがってこのときの Padé 近似式は既約である．

5.7 Padé 近似の事後誤差評価

関数 $f(z)$ の Maclaurin 展開を $n+\nu$ 項で打ち切った z に関する $n+\nu$ 次多項式 F と $G = z^{n+1}$ を与えて，$f(z)$ の Padé 近似式を求める．F と G に Givens 回転による拡張算法 (アルゴリズム B) を適用し，列 (A_i, B_i, C_i) を生成する．ここで，(A_i, B_i, C_i) は拡張剰余列 (P_i, A_i, B_i) を並べ換えて，P_i を C_i で置き換えたものにほかならない．$X_{n+1}(z) = z^{n+1}$ としたときに，(A_i, B_i, C_i) は

$$\begin{aligned}&A_i f - C_i = (A_i f)[X_{n+1}; z] X_{n+1}\\ &\deg A_i + \deg C_i \le n\end{aligned}$$

を満たし，有理式 $C_i(z)/A_i(z)$ は $f(z)$ の Padé 近似式となる．番号 i $(1 \le i \le t)$ において，t 個の Padé 近似式から事後誤差評価によって有理式を選出する．事後誤差評価は $Y = z^{\nu}$ としたときに，$B_i[Y]$ の評価で行う．$B_i[Y]$ は特に

$$B_i[Y] = [B_i]_0^{\nu-1}$$

である．$B_i[Y]$ が小さければ，差分商 $(A_i Qf)[X_{n+1}; z]$ が零に収束したとみなす．ここで，$\deg A_i \ge \deg Q$ である．このとき，例 5.6.1 の評価式 (5.51) から単位円板 $|z| \le 1$ において

$$\begin{aligned}|A_i(z)f(z) - C_i(z)| &= \frac{1}{|Q(z)|}\,|(A_iQf)[X_{n+1}; z]X_{n+1}|\\ &\le \frac{1}{|Q(z)|}\,|(A_iQf)[X_{n+1}; z]|\\ &\le \frac{1}{|Q(z)|}\max_{|\zeta|=R}\frac{\left|[(AQf)(\zeta)]_{n+1}^{\infty}\right|}{R^{n+1}}\frac{1}{1-1/R}\end{aligned}$$

となるので，差分商の評価は単位円板上における Padé 近似の誤差を評価していることになる．上式において，R が 1 より大きいほど Padé 近似式 $C_i(z)/A_i(z)$ の精度は単位円板上において良くなることが分かる．$B_i[Y]$ の大きさは 2 ノルムで計り，

$$\|B_i[Y]\|_2/\gamma' = O(\varepsilon_M)$$

ならば，$B_i[Y]$ は零とみなす．ここで，γ' は

$$\gamma' = \min(1, \gamma),\ \gamma = (\|F\|_1^2 + \|G\|_1^2)^{1/2}$$

で定義された定数，ε_M は計算機イプシロンである．4.4 節の安定性解析にしたがえば，$\|B_i[Y]\|_2/\gamma' = O(\varepsilon_M)$ のとき，$B_i[Y]$ を零とみなしても残差に影響はない．同様に $A_i(z)$ の次数も γ' と比較して $A_i(z)$ の主係数が $O(\varepsilon_M)$ であれば零とみなす．そこで，有理式の選出は次の原則で行う．

$$\begin{aligned}\varepsilon' &= \min_{1\le i\le t}\left\|[B_i]_0^{\nu-1}\right\|_2/\gamma',\\ i_0 &= \min\left(i \mid 1 \le i \le t,\ \left\|[B_i]_0^{\nu-1}\right\|_2/\gamma' \le \max(\varepsilon_M, \varepsilon')\right)\end{aligned} \tag{5.55}$$

この原則によって Padé 近似式の中で $C_{i_0}(z)/A_{i_0}(z)$ を選出する．このときの誤差推定値を

$$\|B_{i_0}[Y]\|_2/\gamma' \tag{5.56}$$

で与える．この原則において，番号 i_0 を小さくとる理由は，有理式の分母 $A_{i_0}(z)$ の次数をできるだけ低次にとることにより $f(z)$ の無縁の極をできるだけ取り込まないためである．分母の理想的な次数は $\deg Q$ である．

5.8 数値例

関数 $f(z)$ の Maclaurin 展開を $n+\nu$ 項で打ち切った多項式 F を Mathematica 32 桁精度で与え，$n=20$, $\nu=2$ として，$f(z)$ の Padé 近似式を HP 715/75 上の Fortran 90 倍精度演算の下で計算した．関数 $f(z)$ の複素平面上における実部 $\mathrm{Re}f(z)$ および 虚部 $\mathrm{Im}f(z)$ の形状を図示した．$f(z)$ に対する F の誤差と式 (5.55) の原則で選出された Padé 近似式の誤差を $x-y$ 平面を複素平面，z 軸を $-\log_{10}(\mathrm{Error})$ としてそれぞれ表示し，特に単位円周上における誤差も表示し，表示の際は $z=e^{i\cdot\theta}$ として θ を 0 から 2π まで動かした．

数値例 5.8.1

$$f(z)=\frac{\tan z}{z}$$

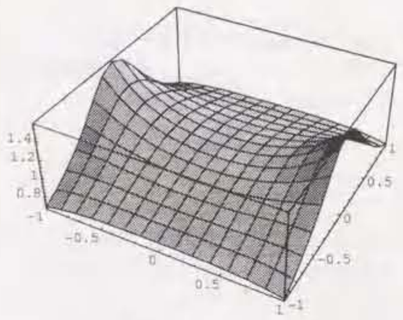

図 5.2: $\mathrm{Re}f(z)$

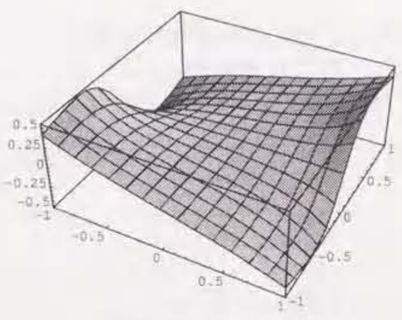

図 5.3: $\mathrm{Im}f(z)$

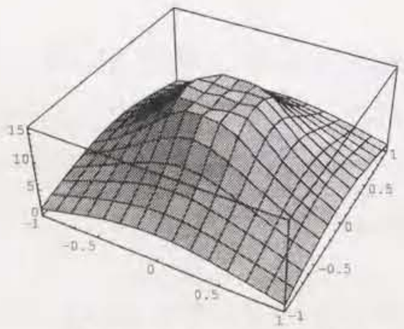

図 5.4: Maclaurin 展開の誤差

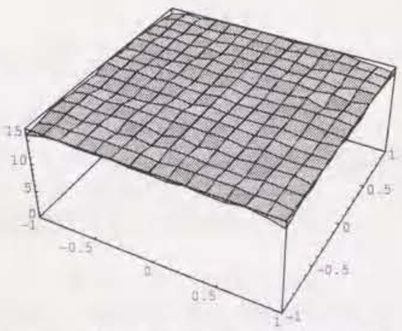

図 5.5: 選出された Padé 近似式の誤差

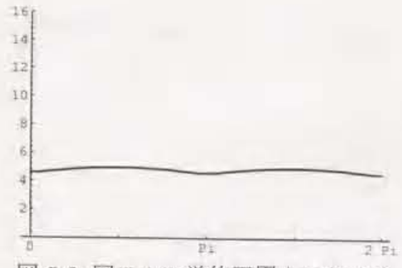

図 5.6: 図 5.4 の単位円周上における誤差

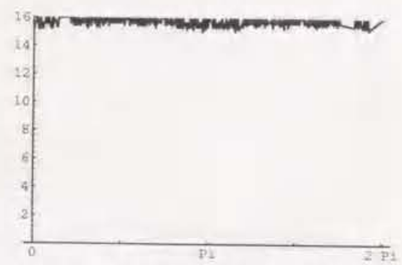

図 5.7: 図 5.5 の単位円周上における誤差

数値例 5.8.2

$$f(z)=\frac{\log(z+3)}{1-2\cos z}$$

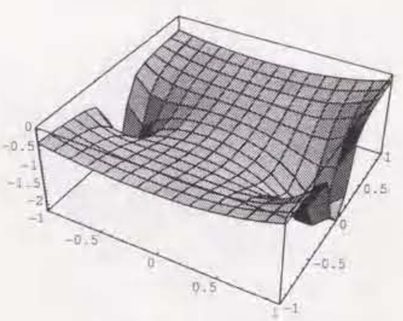

図 5.8: $\mathrm{Re}f(z)$

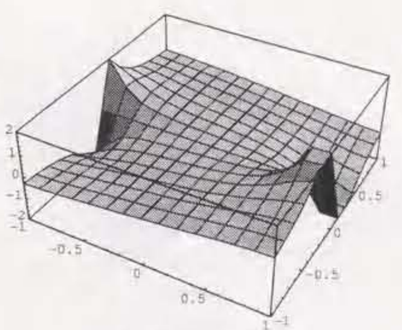

図 5.9: $\mathrm{Im}f(z)$

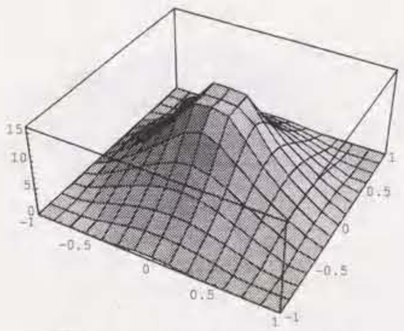

図 5.10: Maclaurin 展開の誤差

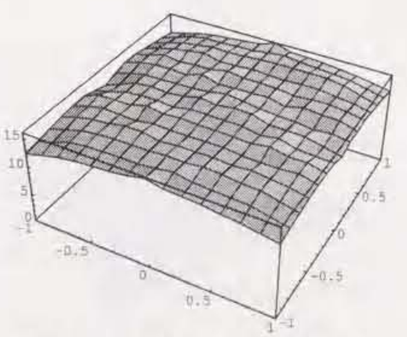

図 5.11: 選出された Padé 近似式の誤差

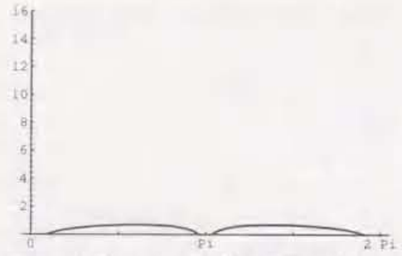

図 5.12: 図 5.10 の単位円周上における誤差

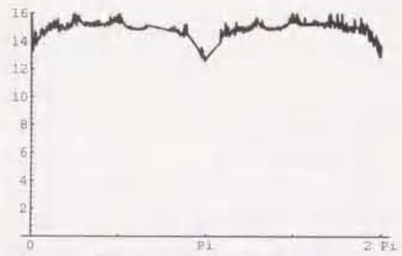

図 5.13: 図 5.11 の単位円周上における誤差

数値例 5.8.3

$$f(z) = \frac{e^z}{(1+z^2)\sqrt{4-z^2}}$$

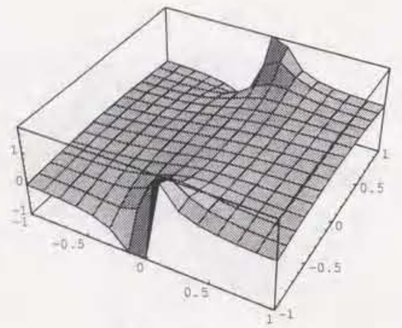

図 5.14: Re$f(z)$

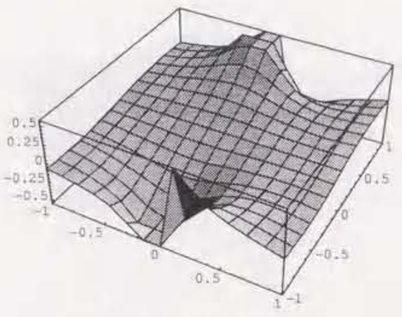

図 5.15: Im$f(z)$

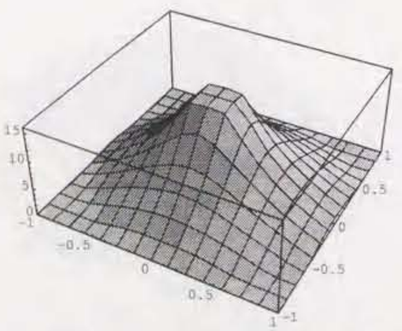

図 5.16: Maclaurin 展開の誤差

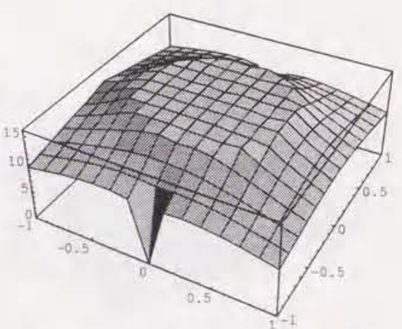

図 5.17: 選出された Padé 近似式の誤差

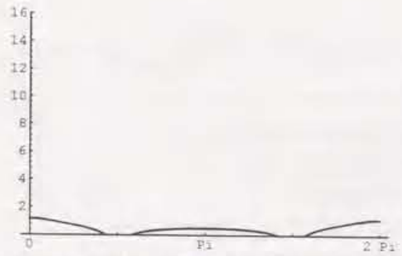

図 5.18: 図 5.16 の単位円周上における誤差

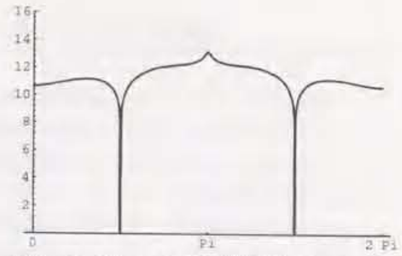

図 5.19: 図 5.17 の単位円周上における誤差

数値例 5.8.1 において，選出された Padé 近似式は，$C_5(z)/A_5(z)$ であり，分子分母の次数はそれぞれ

$$\deg C_5 = 16,\ \deg A_5 = 4.$$

このときの誤差推定値は

$$\left\| [B_5]_0^2 \right\|_2 / \gamma' = 1.11 \times 10^{-18}$$

であった．分母の多項式 A_5 の零点と $f(z)$ の極を表 5.1 に示す．

表 5.1: A_5 の零点と $f(z)$ の極との比較

A_5 の零点	$f(z)$ の極
-4.7128881384707642	-4.7123889803846898
-1.5707963267949290	-1.5707963267948966
$+1.5707963267949290$	$+1.5707963267948966$
$+4.7128881384707642$	$+4.7123889803846898$

この結果を検証してみる．関数 $f(z)$ は無限遠点を除いた領域で有理型である．ここで

$$\begin{aligned} Q_k(z) &= \prod_{j=1}^{k} \left(z + \frac{(2j-1)\pi}{2} \right) \left(z - \frac{(2j-1)\pi}{2} \right) \\ \zeta_{\max}^{(k)} &= (2k-1)\pi/2 \\ g_k(z) &= Q_k(z) f(z) \\ \rho_k &= (2k+1)\pi/2 \end{aligned}$$

とする．$\zeta_{\max}^{(k)}$ は $Q_k(z)$ の零点の絶対値最大値，ρ_k は関数 $g_k(z)$ の原点における収束半径である．ρ_k に対して，$R_k = 0.9 \cdot \rho_k$ にとることにする．R_k は番号 k に関して単調増加するので，k が大きいほど差分商 $(A_5 g_k)[X_{n+1}; z]$ は速く零に収束する．ところで，多項式 $Q_k(z)$ は $\deg Q_k \leq \deg A_5 = 4$ を満たす多項式でなければならないので，番号 k は $k \leq 2$ である．差分商 $(A_5 g_2)[X_{n+1}; z]$ の単位円板上における零への収束の速さ

表 5.2: $A_{13}(z)$ の零点と $f(z)$ の極との比較

$A_{13}(z)$ の零点	$f(z)$ の極
$-6.8577780950477312 - 2.9346893603927882i$	
$-6.8577780950477312 + 2.9346893603927882i$	
$-4.8514722452070318 - 0.5519367559678005i$	
$-4.8514722452070318 + 0.5519367559678005i$	
$-3.5644945694045429 + 0.0000000000000000i$	
$-3.0927364962118120 + 0.0000000000000000i$	
$-1.0471975511966010 + 0.0000000000000000i$	-1.0471975511965977
$+1.0471975511965981 + 0.0000000000000000i$	$+1.0471975511965977$
$+5.2365801666212430 + 0.0000000000000000i$	$+5.2359877559829887$
$+7.2355786405930385 + 0.0000000000000000i$	$+7.3303828583761842$
$+9.3625189890927807 - 2.3040960267221449i$	$+11.519173063162575$
$+9.3625189890927807 + 2.3040960267221449i$	$+17.802358370342161$

は (5.51) より $o(1/R_2^{n+1})$ であり

$$\begin{aligned} 1/R_2^{n+1} &= 1/(0.9 \cdot r_2)^{21} \\ &\approx 1.46 \times 10^{-18} \end{aligned}$$

となる．この数値と数値実験によって得られた誤差推定値とを比較すればほぼ同程度の数値になっている．またこの数値は Padé 近似式 $C_5(z)/A_5(z)$ の単位円板上における精度でもあるので，図 5.5，5.7 において誤差が計算機イプシロン程度になっている．一方，関数 $f(z)$ の A_5 の極への収束の速さは，定理 5.5.3 より $O(\delta^{n+1})$ である．ここで，δ は (5.43) で与えられ

$$\|A_5[Q_k]\| = O\left((\zeta_{\max}^{(k)}/R_k)^{n+1}\right), \quad k = 1, 2$$

となる．上式において $k = 1, 2$ に対してそれぞれ

$$\begin{aligned} \left(\zeta_{\max}^{(1)}/R_1\right)^{n+1} &\approx 8.74 \times 10^{-10}, \\ \left(\zeta_{\max}^{(2)}/R_2\right)^{n+1} &\approx 2.00 \times 10^{-4} \end{aligned}$$

となる．表 5.1において，$Q_1(z)$ の零点に対する $A_5(z)$ の零点の精度は 10 進 13 桁，$Q_2(z)/Q_1(z) = (z + 3\pi/2)(z - 3\pi/2)$ に対する零点の精度は 10 進 4 桁であり，上の 2 式にそれぞれ対応している．

数値例 5.8.2 において，選出された Padé 近似式は，$C_{13}(z)/A_{13}(z)$ であり，分子分母の次数はそれぞれ

$$\deg C_{13} = 8, \ \deg A_{13} = 12.$$

このときの誤差推定値は

$$\left\|[B_{13}]_0^2\right\|_2/\gamma' = 2.36 \times 10^{-16}$$

表 5.3: $A_{13}(z)$ の零点と $f(z)$ の極との比較

$A_{13}(z)$ の零点	$f(z)$ の極
$-2.9634186411899361 + 0.0000000000000000i$	
$-2.2619559526706388 + 0.0000000000000000i$	
$-2.0260203885624328 + 0.0000000000000000i$	
$-1.916004310 \times 10^{-12} - 0.9999999999984051i$	$-1.0000000000000000i$
$-1.916004310 \times 10^{-12} + 0.9999999999984051i$	$+1.0000000000000000i$
$+2.0303252401321501 + 0.0000000000000000i$	
$+2.3112167516200048 + 0.0000000000000000i$	
$+2.8293200965202590 - 5.9382459183767224i$	
$+2.8293200965202590 + 5.9382459183767224i$	
$+3.2592941600080172 + 0.0000000000000000i$	
$+4.3358808912880811 - 2.8508890550284121i$	
$+4.3358808912880811 + 2.8508890550284121i$	

であった．分母の多項式 $A_{13}(z)$ の零点と $f(z)$ の極を表 5.2 に示す．

数値例 5.8.3 において，選出された Padé 近似式は，$C_{13}(z)/A_{13}(z)$ であり，分子分母の次数はそれぞれ

$$\deg C_{13} = 8, \ \deg A_{13} = 12.$$

このときの誤差推定値は

$$\left\|[B_{13}]_0^2\right\|_2/\gamma' = 6.02 \times 10^{-12}$$

であった．分母の多項式 $A_{13}(z)$ の零点と $f(z)$ の極を表 5.3 に示す．

いずれの例も $f(z)$ に対して選出された Padé 近似式の誤差推定値は，ほぼ極を除いた単位円板上における誤差になっている．

5.9 まとめ

関数 $f(z)$ と多項式 $X(z)$ において，$f(z)$ の $X(z)$ における補間多項式 $f[X]$ および差分商 $f[X; z]$ を複素積分表示により表現した．特に差分商の複素積分表示を用いて，有理関数補間の複素領域上における誤差評価を行った．さらに有理補間においては，有理補間式の分母が $f(z)$ の極に収束する速さを示した．有理関数補間において，補間多項式が与えられているとき，有理補間式は，第 3.4，4.3 節の剰余列の拡張算法によって計算される．このとき得られる有理補間式は分子分母の和が一定という条件の下で複数個存在するので，その中で最も良い近似式を選ぶ必要がある．そこで，差分商の評価を利用して，事後誤差評価法を与えた．特に Padé 近似については，単位円板上における誤差評価は差分商の評価に等しい．第 5.7 節では，事後誤差評価法を用いて複数個の Padé 近似式から良い近似式を選出する原則を与えた．数値例では，その原則によって選出された Padé 近似式が，単位円板上においてほぼ誤差推定程度であることを示した．

第 6 章

結論

実または複素係数の多項式剰余列の安定な生成法として，部分終結式による算法と，Givens 回転による算法を提案した．また，これらの算法が有理関数補間問題，代数方程式の解法等に応用できるように，それぞれの拡張算法を提示した．前者の算法の特徴としては，剰余列生成の過程において不正規に近い状態が発生したために起こる桁落ちを抑制するために，パラメータ l を自動的にコントロールして，比較的安定に求めやすい剰余列要素だけを計算するところにある．後者の Givens 回転による算法については，その拡張算法に特徴がある．その特徴の一つとして，拡張算法によって得られた剰余列要素 P の拡張表現 (P, A, B) において，多項式の組 (A, B) が 2 - ノルムの意味で Givens 回転の過程において常に 1 であることである．この性質を利用して，Wilkinson の丸め誤差解析にしたがって，安定性解析を行った．その結果，Givens 回転による方法では，残差の意味において計算機イプシロンのレベルで，拡張剰余列が求まることが分かった．この安定性解析の結果を用いて，従来の方法では困難であった多項式の零判定および主係数の零判定が可能となった．Givens 回転による方法では，それらの零判定の合理的な基準を得ることができた．なおこの算法は従来の方法に比べて計算量が多いので，問題に応じて計算量が自動的に増減するように改善することが今後の課題である．

拡張算法の応用の一つとして，有理関数補間問題がある．当面の目標である関数の自動有理関数近似への布石として，代表的に Padé 近似の場合の事後誤差評価法を提示した．誤差推定値の正当性を理論的に説明し，数値例においてそれを実証した．

本研究の応用としては，他に

- 代数方程式の数値的解法
- 解析関数の零点および極の探索
- 特殊関数等を用いた数値積分変換
- 数値等角写像
- 特異点を考慮した複素平面上における常微分方程式の大域的数値解法 [9, 13, 37]

多変数問題への拡張として

- 多変数の有理関数近似
- 連立高次代数方程式の数値的解法

などがあり，幅広い応用が期待できる．

謝辞

本研究の遂行にあたり，多大なるご指導を賜わりました鳥居達生名古屋大学教授に深甚なる感謝の意を表明します．

本論文の作成にあたり，御教示いだたいた細江繁幸名古屋大学教授，杉原正顯名古屋大学教授に厚くお礼申し上げます．

本論文の重要な構成部分について，長期にわたって直接ご指導いただいた杉浦洋名古屋大学助教授，櫻井鉄也筑波大学助教授に厚くお礼申し上げます．

また，本研究をまとめるにあたり，つねに著者を励まして下さった中島正治鹿児島大学教授に深く感謝申し上げます．

1997 年 4 月
著者

参考文献

[1] Akritas, A.G., Elements of Computer Algebra with Applications, John Wiley & Sons, New York, 1989.

[2] Baker, G. and Graves-Morris, P., Padé Approximants Part I: Basic Theory, Addision-Wesley, Massachusetts, 1981.

[3] Baker, G. and Graves-Morris, P., Padé Approximants Part II: Extensions and Applications, Addision-Wesley, Massachusetts, 1981.

[4] Brezinski, C., Historical Perspective on Interpolation, Approximation and Quadrature, Handbook of Numerical Analysis, P. G. CIARLET and J. L. LIONS · Editors, Vol. III, North-Holland, 1994.

[5] Brown, W.S. and Traub, J.F., On Euclid's Algorithm and the Theory of Subresultants, *J.ACM.18*, (1971), 505-514.

[6] Collins, G.E., Subresultants and Reduced Polynomial Remainder Sequences, *J.ACM.*, Vol.14, No.1 (1967), 128-142.

[7] Corless, R. M., Gianni, P. M., Trager, B. M. and Watt, S. M., The Singular Value Decomposition for Polynomial Systems, Proceedings ISSAC'95, Montreal, 195-207 (1995).

[8] Cuyt, A. and Wuytack, L., Nonlinear Methods in Numerical Analysis, North-Holland, Amsterdam, 1987.

[9] Galiev, V. I, Polupanov, A. F. and Shparlinski, I.E., On the Construction of Solutions of Systems of Linear Ordinary Differential Equations in the Neighbourhood of a Regular Singularity, *J. Comp. Appl. Math.*, Vol. 39, 151-163, (1992).

[10] Gragg, W. B. and Gutknecht, M. H., Stable Look-Ahead Versions of the Euclidean and Chebyshev Algorithms, *International Series of Numerical Mathematics*, Vol. 119, (1994), 231-260.

[11] Gutknecht, M. H. and Hochbruck, M., Optimized Look-Ahead Recurrences for Adjacent Rows in the Padé Table, *BIT 36:2* (1996), 264-286.

[12] Henrici, P., Applied and Computational Complex Analysis, Vol. 1, Power Series-Integration-Conformal Mapping-Location of Zeros, John Wiley & Sons, New York, 1974.

[13] Henrici, P., Applied and Computational Complex Analysis, Vol. 2, Special Functions-Integral Transforms-Asymptotics-Continued Fractions, John Wiley & Sons, New York, 1977.

[14] 東田幸樹，山本芳人，熊沢友信，入門 Fortran 90 実践プログラミング，ソフトバンク，東京，1994.

[15] Metcalf, M. and Reid, J. 著，西村恕彦，和田英穂，西村和夫，高田正之訳，bit 別冊詳細 Fortran 90，共立出版，東京，1993.

[16] 森正武，数値解析と複素関数論，数理科学シリーズ 7，筑摩書房，東京，1975.

[17] 中溝高好，線形離散時間システムの同定手法 IV.，システムと制御，Vol.25, No.12, (1981), 755-767.

[18] 宮広栄一，野田松太郎，新しい有理関数近似によるハイブリッド積分の拡張について，日本応用数理学会論文誌，Vol.2, No.4 (1992), 193-206.

[19] 大迫尚行，櫻井鉄也，杉浦洋，鳥居達生，多項式剰余列の安定な生成法，日本応用数理学会論文誌，Vol. 5, No. 3, (1995), 241–255.

[20] 大迫尚行，杉浦洋，鳥居達生，Givens 回転による多項式剰余列の拡張算法，情報処理学会論文誌，Vol. 38, No. 1, (1997), 158-160.

[21] 大迫尚行，杉浦洋，鳥居達生，多項式剰余列の安定な拡張算法，日本応用数理学会論文誌（投稿中）.

[22] Oppenheim, A. V. and Schafer, R. W. 著，伊達玄 訳，ディジタル信号処理（上），コロナ社，東京，1978.

[23] Pozzi, A., Applications of Padé Approximation Theory in Fruid Dynamics, World Scientific, Singapore, 1994.

[24] ロベルト ヴィーフ著，富久泰明監訳，松下電工（株）訳，電子/制御/システム工学のための Z 変換の理論と応用，理工学海外名著シリーズ 47，プレイン図書出版，愛媛，1991.

[25] 櫻井鉄也，補間法の拡張を応用した代数方程式の分割統治解法に関する研究，博士学位論文，名古屋大学，1992.

[26] Sakurai, T., Torii, T. and Sugiura, H., An Iterative Method for Algebraic Equation by Padè Approximation, *Computing 46* (1991), 131-141.

[27] Sakurai, T., Sugiura, H. and Torii, T., Numerical Factorization of a Polynomial by Rational Hermite Interpolation, *Numerical Algorithms 3* (1992), 411-418.

[28] Sasaki, T. and Noda, M., Approximate Square-free Decomposition and Root-finding of Ill-conditioned Algebraic Equations, *J. Inf. Process.*, Vol.12, No.2 (1989), 159-168.

[29] Sasaki, T. and Sasaki, M., Analysis of Accuracy Decreasing in Polynomial Remainder Sequence with Floating-point Number Coefficients, *J. Inf. Process.*, Vol.12, No.4 (1989), 384-403.

[30] Sasaki, T., A Study of Approximate Polynomials, I — Representation and Arithmetic —, *Japan J. Indust. Appl. Math.*, 12 (1995), 137–161.

[31] 佐々木建昭，今井浩，浅野孝夫，杉原厚吉，計算代数と計算幾何，岩波講座 応用数学 5，岩波書店，東京，1993.

[32] 佐々木建昭，数式処理 情報処理叢書 7，情報処理学会，東京，1981.

[33] 園田信吾，櫻井鉄也，杉浦洋，鳥居達生，分割統治法による多項式の数値的因数分解，日本応用数理学会論文誌，Vol.1, No.4 (1991), 277-290.

[34] 杉浦洋，FFT の一般化とその数値解析的応用に関する研究，博士学位論文，名古屋大学，1991.

[35] 高木貞治，初等整数論講義第 2 版，共立出版，東京，1971.

[36] Torii, T., Sakurai, T. and Sugiura, H., An Application of Sunzi's Theorem for Solving Algebraic Equations, *Proceeding of the First China-Japan Seminar on Numerical Mathematics, Beijing 1992, Series on Applied Math.*, Vol.5, 155-167, World Scientific, Singapore, 1993.

[37] Wasow, W., Asymptotic Expansions for Ordinary Differential Equations, Dover, New York, 1965.

[38] Wilkinson, J. H., The Algebraic Eigenvalue Problem, Oxford University Press, London, 1965.

付録

A Fortran 90 プログラム

多項式に関する数値計算プログラムを Fortran 90 で記述した．ここでは，実係数の多項式を取り扱う．複素係数については，このプログラムにおいて，real を complex で置き換えるだけで複素係数版のモジュールとなり，本質的には実係数版と同じであるので省略する．プログラム単位モジュールを次の 5 つに分類した．

1. module precision（精度を与えるパラメータを定義）
2. module polynomial_system（多項式に関する演算のモジュール）
3. module triplet_system（多項式 3 つ組に関する演算のモジュール）
4. module polynomial_algorithm（多項式のアルゴリズムに関するモジュール）
5. module ex_polynomial_algorithm（多項式 3 つ組のアルゴリズムに関するモジュール）

モジュール名 module precision において，精度を与える整数型パラメータ kd を種別関数 selected_real_kind を用いて定義した．この種別関数の引数として整数型定数 15 を与え，実数の取り扱う精度を 15 桁以上（倍精度演算）にした．あえて種別関数を用いて定義する理由は，種別型パラメータ値が機種によって異なる場合があるからである．例えば，倍精度に対する種別型パラメータ値は，機種によっては 2 であったり，バイト数の 8 であったりする．単精度，四倍精度演算を行いたい場合には，種別関数の引数を 6 または 32 と変更すればよい．引数の変更の度に module precision を引用しているモジュールを再コンパイルする必要があるが，その際，各モジュールの包含関係に注意して makefile を作成する必要がある．あらかじめ単精度，倍精度，四倍精度版のモジュールをそれぞれ作成しておけば，再コンパイルを労しない．複数の手続き名を，引数の型あるいは引数の個数に応じて同じ手続き名で使いたい場合には，interface 文でそれらの手続き名を登録しておけばよい．例えば，モジュール名 module polynomial_system において，多項式から多項式への代入文を定義する手続き名 poly_to_poly と定数から多項式への代入文を定義する手続き名 constant_to_poly を同じ記号 = で使用するために interface assignment(=) で多重定義している．

A.1 多項式に関する演算のモジュール

z に関する n 次多項式 P の係数は

$$P = p_n z^n + p_{n-1} z^{n-1} + \cdots + p_0$$

の順に対応している．

```
module precision
  ! 精度を与えるパラメータ
  integer,parameter :: kd=selected_real_kind(15)
end module precision

module polynomial_system
  ! 多項式演算の module
  use precision
  type polynomial
    ! 多項式の型の定義
    integer :: deg
    real(kind=kd),dimension(:),pointer :: coef
  end type polynomial

  interface operator(+)
    ! 多項式の加算
    module procedure add_polynomial
  end interface

  interface operator(-)
    ! 多項式の減算，-（多項式）
    module procedure sub_polynomial
    module procedure sgn_minus_polynomial
  end interface

  interface operator(*)
    ! 多項式の乗算，乗数倍
    module procedure mult_polynomial
    module procedure mult_scaler
  end interface

  interface operator(/)
    ! 多項式の除算（商），多項式/定数
    module procedure div_polynomial
```

```
  module procedure div_scaler
end interface

interface operator(.mod.)
  ! 多項式の除算（剰余）
  module procedure mod_polynomial
end interface

interface assignment(=)
  ! 代入文
  module procedure poly_to_poly
  ! 多項式 = 多項式
  module procedure constant_to_poly
  ! 多項式 = 定数
end interface

interface Elim
  ! 主係数消去演算 Elim
  module procedure Elim_poly
end interface

interface Piv
  ! 枢軸選び操作 Piv
  module procedure Piv_poly
end interface

interface Shift
  ! 多項式の z のべき乗によるシフト演算
  module procedure Shift_poly
end interface

interface Givens
  ! 主係数消去演算 Givens
  module procedure Givens_poly
end interface

interface print_poly
  ! 多項式の出力
  module procedure print_poly
end interface
```



```
interface value
  ! 多項式のある点における値
  module procedure value
  ! 実数値における値
  module procedure value_complex
  ! 複素数値における値
end interface

contains

  type(polynomial) function add_polynomial(p,q) result(r)
    ! 多項式の加算
    type(polynomial),intent(in) :: p,q
    integer :: i
    if (p%deg==q%deg) then
      do i=p%deg,1,-1
        if (p%coef(i)/=-q%coef(i)) exit
      end do
      r%deg=i
      allocate(r%coef(0:r%deg))
      r%coef=p%coef(0:r%deg)+q%coef(0:r%deg)
    else
      r%deg=max(p%deg,q%deg)
      allocate(r%coef(0:r%deg))
      if (p%deg==r%deg) then
        r%coef=p%coef
        r%coef(0:q%deg)=r%coef(0:q%deg)+q%coef
      else
        r%coef=q%coef
        r%coef(0:p%deg)=r%coef(0:p%deg)+p%coef
      end if
    end if
  end function add_polynomial

  type(polynomial) function sub_polynomial(p,q) result(r)
    ! 多項式の減算
    type(polynomial),intent(in) :: p,q
    integer :: i
    if (p%deg==q%deg) then
```

```
      do i=p%deg,1,-1
        if (p%coef(i)/=q%coef(i)) exit
      end do
      r%deg=i
      allocate(r%coef(0:r%deg))
      r%coef=p%coef(0:r%deg)-q%coef(0:r%deg)
    else
      r%deg=max(p%deg,q%deg)
      allocate(r%coef(0:r%deg))
      if (p%deg==r%deg) then
        r%coef=p%coef
        r%coef(0:q%deg)=r%coef(0:q%deg)-q%coef
      else
        r%coef=-q%coef
        r%coef(0:p%deg)=r%coef(0:p%deg)+p%coef
      end if
    end if
end function sub_polynomial

type(polynomial) function sgn_minus_polynomial(p) result(r)
  ! -（多項式）
  type(polynomial),intent(in) :: p
  r%deg=p%deg
  allocate(r%coef(0:r%deg))
  r%coef=-p%coef
end function sgn_minus_polynomial

type(polynomial) function mult_polynomial(p,q) result(r)
  ! 多項式の乗算
  type(polynomial),intent(in) :: p,q
  integer :: i,j
  logical :: z
  z=p%deg==0.and.p%coef(0)==0.or.q%deg==0.and.q%coef(0)==0
  if (z) then
    r=0.0_kd
  else
    r%deg=p%deg+q%deg
    allocate(r%coef(0:r%deg))
    r%coef=0
    do i=0,p%deg
```



```
      do j=0,q%deg
        r%coef(i+j)=r%coef(i+j)+p%coef(i)*q%coef(j)
      end do
    end do
  end if
end function mult_polynomial

type(polynomial) function mult_scaler(c,p) result(r)
  ! 多項式の乗数（実数）倍
  real(kind=kd),intent(in) :: c
  type(polynomial),intent(in) :: p
  r%deg=p%deg
  allocate(r%coef(0:r%deg))
  r%coef=c*p%coef
end function mult_scaler

type(polynomial) function div_polynomial(p,q) result(r)
  ! 多項式の除算（商）
  type(polynomial),intent(in) :: p,q
  real(kind=kd) :: mult
  integer :: i
  logical :: z
  z=q%deg==0.and.q%coef(0)==0
  if (z) then
    write(*,*)'zero divided'
    r=0.0_kd
  elseif (q%deg==0) then
    r=p/q%coef(0)
  else if (p%deg<q%deg) then
    r=0.0_kd
  else
    r=p
    do i=p%deg-q%deg,0,-1
      mult=r%coef(i+q%deg)/lc(q)
      r%coef(i:i+q%deg-1)= &
      & r%coef(i:i+q%deg-1)-mult*q%coef(0:q%deg-1)
      r%coef(i+q%deg)=mult
    end do
    r%deg=p%deg-q%deg
    r%coef(0:r%deg)=r%coef(q%deg:p%deg)
```

```
  end if
end function div_polynomial

type(polynomial) function div_scaler(p,c) result(r)
  ! 多項式/定数
  type(polynomial),intent(in) :: p
  real(kind=kd),intent(in) :: c
  r%deg=p%deg
  allocate(r%coef(0:r%deg))
  r%coef=p%coef/c
end function div_scaler

type(polynomial) function mod_polynomial(p,q) result(r)
  ! 多項式の除算（剰余）
  type(polynomial),intent(in) :: p,q
  integer :: i
  logical :: z
  z=q%deg==0.and.q%coef(0)==0
  if (z) then
    write(*,*)'zero divided'
    r=0.0_kd
  else if (q%deg==0) then
    r=0.0_kd
  else if (p%deg<q%deg) then
    r=p
  else
    r=p
    do i=p%deg-q%deg,0,-1
      r%coef(i:i+q%deg-1)=r%coef(i:i+q%deg-1) &
      & -r%coef(i+q%deg)/lc(q)*q%coef(0:q%deg-1)
    end do
    do i=q%deg-1,1,-1
      if (r%coef(i)/=0) exit
    end do
    r%deg=i
  endif
end function mod_polynomial

subroutine poly_to_poly(p,q)
  ! 代入文
```



```
  ! 多項式 = 多項式
  type(polynomial),intent(in) :: q
  type(polynomial),intent(inout) :: p
  real(kind=kd),dimension(:),allocatable :: w
  allocate(w(0:q%deg))
  w=q%coef
  p%deg=q%deg
  allocate(p%coef(0:p%deg))
  p%coef=w
  deallocate(w)
end subroutine poly_to_poly

subroutine constant_to_poly(p,c)
  ! 代入文
  ! 多項式 = 定数
  real(kind=kd),intent(in) :: c
  type(polynomial),intent(inout) :: p
  p%deg=0
  allocate(p%coef(0:p%deg))
  p%coef=c
end subroutine constant_to_poly

type(polynomial) function Elim_poly(f,g) result(r)
  ! 主係数消去演算 Elim
  type(polynomial) :: f,g
  real(kind=kd) :: p
  r%deg=f%deg-1
  allocate(r%coef(0:r%deg))
  if (abs(lc(f)) > abs(lc(g))) then
    p=lc(g)/lc(f)
    r%coef(0:r%deg)=g%coef(0:r%deg)-p*f%coef(0:r%deg)
  else
    p=lc(f)/lc(g)
    r%coef(0:r%deg)=f%coef(0:r%deg)-p*g%coef(0:r%deg)
  end if
end function Elim_poly

subroutine Piv_poly(f,ip,i0,i1) !多項式列の枢軸選び
  type(polynomial),dimension(0:i1),intent(in) :: f
  integer,intent(in) :: i0,i1
```

```
  integer,dimension(0:i1),intent(inout) :: ip
  dimension imax(1)
  integer :: i,iw
  imax=maxloc((/(abs(lc(f(ip(i)))),i=i0,i1)/))+i0-1
  if (imax(1)==i0) return
  iw=ip(i0)
  ip(i0)=ip(imax(1))
  ip(imax(1))=iw
end subroutine Piv_poly

type(polynomial) function Shift_poly(p,d) result(r)
  ! 多項式の z のべき乗によるシフト演算
  type(polynomial),intent(in) :: p
  integer,intent(in) :: d
  if (p%deg==0.and.p%coef(0)==0.or.d==0) then
    r=p
  else
    r%deg=p%deg+d
    allocate(r%coef(0:r%deg))
    r%coef(0:d-1)=0
    r%coef(d:)=p%coef
  end if
end function Shift_poly

subroutine Givens_poly(f,g,f1,g1)
  ! 主係数消去演算 Givens (Givens 回転)
  type(polynomial),intent(in) :: f,g
  type(polynomial),intent(inout) :: f1,g1
  type(polynomial),dimension(:),allocatable :: w
  real(kind=kd) :: c,s,d
  if (lc(g)/=0) then
    allocate(w(2))
    d=sqrt_sum_2(lc(f),lc(g))
    c=lc(f)/d ; s=-lc(g)/d
    w(1)%deg=f%deg ; w(2)%deg=g%deg-1
    allocate(w(1)%coef(0:w(1)%deg),w(2)%coef(0:w(2)%deg))
    w(1)%coef=c*f%coef(0:w(1)%deg)-s*g%coef(0:w(1)%deg)
    w(2)%coef=s*f%coef(0:w(2)%deg)+c*g%coef(0:w(2)%deg)
    f1=w(1) ; g1=w(2)
    deallocate(w)
```



```
  else
    f1=f ; g1=g ; g1%deg=g1%deg-1
  endif
end subroutine Givens_poly

type(polynomial) function series_rational(p,q,n) result(r)
  ! 有理式 p/q を n 次多項式に変換する.
  type(polynomial) :: p,q
  integer :: j,k,m,n,pdeg,qdeg
  if (q%coef(0)==0) then
    write(*,*)'illegal q(0)=0!'
    r=q
    return
  else if (q%deg==0) then
    r%deg=min(n,p%deg)
    allocate(r%coef(0:r%deg))
    if (q%coef(0)==1) then
      r%coef=p%coef(0:r%deg)
    else
      r%coef=p%coef(0:r%deg)/q%coef(0)
    end if
    return
  end if
  r%deg=n
  allocate(r%coef(0:r%deg))
  pdeg=min(p%deg,n) ; qdeg=min(q%deg,n)
  r%coef(0)=p%coef(0)/q%coef(0)
  do k=1,min(pdeg,qdeg)
    r%coef(k)= &
    & (p%coef(k) &
    & -sum((/(q%coef(k-j)*r%coef(j),j=0,k-1)/)))/q%coef(0)
  end do
  if (pdeg>=qdeg) then
    do k=qdeg+1,pdeg
      r%coef(k)= &
      & (p%coef(k)-sum((/(q%coef(qdeg-j)*r%coef(k-qdeg+j) &
      & ,j=0,qdeg-1)/)))/q%coef(0)
    end do
  else
    do k=pdeg+1,qdeg
```

```
      r%coef(k)= &
      & -sum((/(q%coef(k-j)*r%coef(j),j=0,k-1)/))/q%coef(0)
    end do
  end if
  m=max(pdeg,qdeg)
  do k=m+1,n
    r%coef(k)= &
    & -sum((/(q%coef(qdeg-j)*r%coef(k-qdeg+j) &
    & ,j=0,qdeg-1)/))/q%coef(0)
  end do
end function series_rational

type(polynomial) function exp_p(n) result(r)
  ! 指数関数の Maclaurin 展開を n 項で打ち切った多項式
  integer :: k,n
  r%deg=n
  allocate(r%coef(0:r%deg))
  r%coef(0)=1
  do k=1,n
    r%coef(k)=r%coef(k-1)/k
  end do
end function exp_p

type(polynomial) function sin_p(n) result(r)
  ! sine 関数の Maclaurin 展開を n 項で打ち切った多項式
  integer :: k,n
  r%deg=n-mod(n+1,2)
  allocate(r%coef(0:r%deg))
  r%coef=0
  r%coef(1)=1
  do k=3,r%deg,2
    r%coef(k)=-r%coef(k-2)/(k*(k-1))
  end do
end function sin_p

type(polynomial) function cos_p(n) result(r)
  ! cosine 関数の Maclaurin 展開を n 項で打ち切った多項式
  integer :: k,n
  r%deg=n-mod(n,2)
  allocate(r%coef(0:r%deg))
```



```
  r%coef=0
  r%coef(0)=1
  do k=2,r%deg,2
    r%coef(k)=-r%coef(k-2)/(k*(k-1))
  end do
end function cos_p

subroutine q_and_mod(f,g,q,r)
  ! 多項式 f を g で割ったときの 商 q と 剰余 r を求める.
  type(polynomial),intent(in) :: f,g
  type(polynomial),intent(out) :: q,r
  real(kind=kd) :: mult
  integer :: i
  logical :: z
  z=g%deg==0.and.g%coef(0)==0
  if (z) then
    write(*,*)'zero divided'
    q=0.0_kd ; r=0.0_kd
  elseif (g%deg==0) then
    q=f/g%coef(0) ; r=0.0_kd
  else if (f%deg<g%deg) then
    q=0.0_kd ; r=f
  else
    q%deg=f%deg-g%deg
    allocate(q%coef(0:q%deg))
    r=f
    do i=q%deg,0,-1
      mult=r%coef(i+g%deg)/lc(g)
      r%coef(i:i+g%deg-1)= &
      & r%coef(i:i+g%deg-1)-mult*g%coef(0:g%deg-1)
      r%coef(i+g%deg)=mult
    end do
    q%coef=r%coef(g%deg:f%deg)
    do i=g%deg-1,1,-1
      if (r%coef(i)/=0) exit
    end do
    r%deg=i
  end if
end subroutine q_and_mod
```

```
subroutine stable_q_and_mod(f,g,q,r,mu)
  ! 安定な割り算ルーチン
  type(polynomial),intent(in) :: f,g
  type(polynomial),intent(out) :: q,r
  real(kind=kd),intent(out) :: mu
  real(kind=kd) :: nu
  integer :: d,i
  logical :: z
  z=g%deg==0.and.g%coef(0)==0
  if (z) then
    write(*,*)'zero divided'
    q=0.0_kd ; r=0.0_kd
  elseif (g%deg==0) then
    q=f/g%coef(0) ; r=0.0_kd
  else if (f%deg<g%deg) then
    q=0.0_kd ; r=f
  else
    q%deg=f%deg-g%deg
    allocate(q%coef(0:q%deg))
    r=f
    mu=1
    do while(r%deg>=g%deg)
      d=r%deg-g%deg
      if (abs(lc(r))<=abs(lc(g))) then
        nu=lc(r)/lc(g)
        r%coef(d:r%deg-1)=r%coef(d:r%deg-1)-nu*g%coef(0:g%deg-1)
        q%coef(d)=nu
      else
        nu=lc(g)/lc(r)
        r%coef(d:r%deg-1)=nu*r%coef(d:r%deg-1)-g%coef(0:g%deg-1)
        q%coef(d)=1
        q%coef(d+1:q%deg)=nu*q%coef(d+1:q%deg)
        mu=mu*nu
      end if
      if (d>0) r%coef(d-1)=mu*r%coef(d-1)
      r%deg=r%deg-1
    end do
    do i=g%deg-1,1,-1
      if (r%coef(i)/=0) exit
    end do
```



```
    r%deg=i
  end if
end subroutine stable_q_and_mod

function value(p,z)
  ! 多項式 p の z における値
  real(kind=kd) :: value,z
  type(polynomial) :: p
  integer :: i
  value=lc(p)
  do i=p%deg-1,0,-1
    value=p%coef(i)+z*value
  end do
end function value

function value_complex(p,z) result(r)
  ! 多項式 p の z における値 (z は 複素数)
  complex(kind=kd) :: r,z
  type(polynomial) :: p
  integer :: i
  r=lc(p)
  do i=p%deg-1,0,-1
    r=p%coef(i)+z*r
  end do
end function value_complex

type(polynomial) function D_poly(p) result(r)
  ! 多項式 p の 導関数
  type(polynomial) :: p
  integer :: i
  r%deg=p%deg-1
  allocate(r%coef(0:r%deg))
  do i=0,r%deg
    r%coef(i)=(i+1)*p%coef(i+1)
  end do
end function D_poly

subroutine exact_deg(delta,p)
  ! 多項式 p の真の次数の決定
  real(kind=kd),intent(in) :: delta
```

```
    type(polynomial),intent(inout) :: p
    do while(abs(lc(p))<=delta.and.p%deg>=0)
      p%deg=p%deg-1
    end do
  end subroutine exact_deg

  function norm(p,m)
    ! 多項式 p の無限大ノルム
    real(kind=kd) :: norm
    type(polynomial) :: p
    integer,optional :: m
    integer :: i
    if (present(m)) then
      norm=maxval((/(abs(p%coef(i)),i=0,m-1)/))
    else
      norm=maxval((/(abs(p%coef(i)),i=0,p%deg)/))
    end if
  end function norm

  function norm_1(p,m)
    ! 多項式 p の 1 ノルム
    type(polynomial),intent(in) :: p
    integer,optional,intent(in) :: m
    real(kind=kd) :: norm_1
    integer :: i
    if (present(m)) then
      norm_1=sum((/(abs(p%coef(i)),i=0,m-1)/))
    else
      norm_1=sum((/(abs(p%coef(i)),i=0,p%deg)/))
    end if
  end function norm_1

  function norm_2(p,m)
    ! 多項式 p の 2 ノルム
    type(polynomial),intent(in) :: p
    integer,optional,intent(in) :: m
    real(kind=kd) :: norm_2
    integer :: i
    if (present(m)) then
      norm_2=sum((/(p%coef(i)**2,i=0,m-1)/))
```



```
    else
      norm_2=sum((/(p%coef(i)**2,i=0,p%deg)/))
    end if
    norm_2=sqrt(norm_2)
  end function norm_2

  function sqrt_sum_2(a,b) result(r)
    ! 2 数 a, b の絶対値二乗和平方
    real(kind=kd) :: a,b,r,abs_a,abs_b
    abs_a=abs(a) ; abs_b=abs(b)
    r=max(abs_a,abs_b)
    if (r==0) return
    if (r==abs_a) then
      r=r*sqrt(1+(abs_b/r)**2)
    else
      r=r*sqrt(1+(abs_a/r)**2)
    end if
  end function sqrt_sum_2

  function lc(f)
    ! 多項式 f の主係数
    real(kind=kd) :: lc
    type(polynomial) :: f
    lc=f%coef(f%deg)
  end function lc

  subroutine print_polynomial(p)
    ! 多項式 p の出力ルーチン
    type(polynomial),intent(in) :: p
    integer :: i
    if (p%deg==0) then
      write(*,*) p%coef(0)
      return
    end if
    write(*,*)'(',p%coef(0),')+'
    do i=1,p%deg-1
      write(*,*)'(',p%coef(i),') z^',i,'+'
    end do
    write(*,*)'(',lc(p),') z^',p%deg
  end subroutine print_polynomial
```

```
  subroutine print_poly(p,n)
    ! 多項式 p の出力ルーチン
    type(polynomial),intent(in) :: p
    integer,optional,intent(in) :: n
    write(*,*)
    if (present(n)) then
      write(*,*)'p(',n,')(z)='
      call print_polynomial(p)
    else
      write(*,*)'p(z)='
      call print_polynomial(p)
    end if
  end subroutine print_poly

end module polynomial_system
```

A.2 多項式 3 つ組に関する演算のモジュール

```
module triplet_system
  use polynomial_system
  type triplet_poly
    ! 多項式 3 つ組の型の定義
    type(polynomial) :: a,b,c
  end type triplet_poly

  interface operator(+)
    ! 多項式 3 つ組の加算
    module procedure add_triplet_poly
  end interface

  interface operator(-)
    ! 多項式 3 つ組の減算, -(多項式 3 つ組)
    module procedure sub_triplet_poly
    module procedure sgn_minus_triplet_poly
  end interface

  interface operator(*)
    ! 多項式 * (多項式 3 つ組), 乗数倍
    module procedure mult_triplet_poly
```



```
    module procedure mult_scaler_triplet_poly
  end interface

  interface operator(/)
    ! (多項式 3 つ組)/多項式,(多項式 3 つ組)/定数
    module procedure div_triplet_poly
    module procedure div_scaler_triplet_poly
  end interface

  interface assignment(=)
    ! 代入文
    module procedure triplet_to_triplet
    ! 多項式 3 つ組 = 多項式 3 つ組
  end interface

  interface Elim
    ! 主係数消去演算 Elim
    module procedure Elim_triplet
  end interface

  interface Piv
    ! 枢軸選び操作 Piv
    module procedure Piv_triplet
  end interface

  interface Shift
    module procedure Shift_triplet
  end interface

  interface Givens
    ! 多項式 3 つ組の z のべき乗によるシフト演算
    module procedure Givens_triplet
  end interface

  interface print_poly
    ! 多項式 3 つ組の出力
    module procedure print_triplet
  end interface

contains
```

```
type(triplet_poly) function add_triplet_poly(p,q) result(r)
  ! 多項式 3 つ組の加算
  type(triplet_poly),intent(in) :: p,q
  r%a=p%a+q%a
  r%b=p%b+q%b
  r%c=p%c+q%c
end function add_triplet_poly

type(triplet_poly) function sub_triplet_poly(p,q) result(r)
  ! 多項式 3 つ組の減算
  type(triplet_poly),intent(in) :: p,q
  r%a=p%a-q%a
  r%b=p%b-q%b
  r%c=p%c-q%c
end function sub_triplet_poly

type(triplet_poly) function sgn_minus_triplet_poly(p) result(r)
  ! -（多項式 3 つ組）
  type(triplet_poly),intent(in) :: p
  r%a=-p%a
  r%b=-p%b
  r%c=-p%c
end function sgn_minus_triplet_poly

type(triplet_poly) function mult_triplet_poly(q,p) result(r)
  ! 多項式 * （多項式 3 つ組）
  type(triplet_poly),intent(in) :: p
  type(polynomial),intent(in) :: q
  r%a=q*p%a
  r%b=q*p%b
  r%c=q*p%c
end function mult_triplet_poly

type(triplet_poly) function mult_scaler_triplet_poly(c,p) &
  & result(r)
  ! 多項式 3 つ組の乗数倍
  type(triplet_poly),intent(in) :: p
  real(kind=kd),intent(in) :: c
  r%a=c*p%a
```



```
  r%b=c*p%b
  r%c=c*p%c
end function mult_scaler_triplet_poly

type(triplet_poly) function div_triplet_poly(p,q) result(r)
  ! （多項式 3 つ組）/多項式
  type(triplet_poly),intent(in) :: p
  type(polynomial),intent(in) :: q
  r%a=p%a/q
  r%b=p%b/q
  r%c=p%c/q
end function div_triplet_poly

type(triplet_poly) function div_scaler_triplet_poly(p,c) &
  & result(r)
  ! （多項式 3 つ組）/定数
  type(triplet_poly),intent(in) :: p
  real(kind=kd),intent(in) :: c
  r%a=p%a/c
  r%b=p%b/c
  r%c=p%c/c
end function div_scaler_triplet_poly

subroutine triplet_to_triplet(p,q)
  ! 代入文
  ! 多項式 3 つ組 = 多項式 3 つ組
  type(triplet_poly),intent(in) :: q
  type(triplet_poly),intent(inout) :: p
  type(triplet_poly) :: w
  w%a=q%a ; w%b=q%b ; w%c=q%c
  p%a=w%a ; p%b=w%b ; p%c=w%c
end subroutine triplet_to_triplet

type(triplet_poly) function Elim_triplet(f,g) result(r)
  ! 主係数消去演算 Elim
  type(triplet_poly) :: f,g
  real(kind=kd) :: p
  r%c%deg=f%c%deg-1
  allocate(r%c%coef(0:r%c%deg))
  if (abs(lc(f%c)) > abs(lc(g%c))) then
```

```
    p=lc(g%c)/lc(f%c)
    r%c%coef(0:r%c%deg)=g%c%coef(0:r%c%deg)-p*f%c%coef(0:r%c%deg)
    r%a=g%a-p*f%a
    r%b=g%b-p*f%b
  else
    p=lc(f%c)/lc(g%c)
    r%c%coef(0:r%c%deg)=f%c%coef(0:r%c%deg)-p*g%c%coef(0:r%c%deg)
    r%a=f%a-p*g%a
    r%b=f%b-p*g%b
  end if
end function Elim_triplet

subroutine Givens_triplet(f,g,f1,g1)
  ! 主係数消去演算 Givens （Givens 回転）
  type(triplet_poly),intent(in) :: f,g
  type(triplet_poly),intent(inout) :: f1,g1
  type(triplet_poly),dimension(:),allocatable :: w
  real(kind=kd) :: c,s,d
  if (lc(g%c)/=0) then
    allocate(w(2))
    d=sqrt_sum_2(lc(f%c),lc(g%c))
    c=lc(f%c)/d ; s=-lc(g%c)/d
    w(1)%c%deg=f%c%deg ; w(2)%c%deg=g%c%deg-1
    allocate(w(1)%c%coef(0:w(1)%c%deg),w(2)%c%coef(0:w(2)%c%deg))
    w(1)%c%coef=c*f%c%coef(0:w(1)%c%deg)-s*g%c%coef(0:w(1)%c%deg)
    w(1)%a=c*f%a-s*g%a ; w(1)%b=c*f%b-s*g%b
    w(2)%c%coef=s*f%c%coef(0:w(2)%c%deg)+c*g%c%coef(0:w(2)%c%deg)
    w(2)%a=s*f%a+c*g%a ; w(2)%b=s*f%b+c*g%b
    f1=w(1) ; g1=w(2)
    deallocate(w)
  else
    f1=f ; g1=g ; g1%c%deg=g1%c%deg-1
  endif
end subroutine Givens_triplet

type(triplet_poly) function Shift_triplet(f,d) result(r)
  ! 多項式 3 つ組の z のべき乗によるシフト演算
  type(triplet_poly),intent(in) :: f
  integer,intent(in) :: d
  r%a=Shift(f%a,d)
```



```
  r%b=Shift(f%b,d)
  r%c=Shift(f%c,d)
end function Shift_triplet

function zansa(f,g,p)
  ! 残差 r=a f+b g-p の 1 ノルム
  type(polynomial) :: f,g
  type(triplet_poly) :: p
  real(kind=kd) :: zansa
  zansa=norm_1(p%a*f+p%b*g-p%c)
end function zansa

function zansa_2(f,g,p)
  ! 残差 r=a f+b g-p の 2 ノルム
  type(polynomial) :: f,g
  type(triplet_poly) :: p
  real(kind=kd) :: ab,fg,zansa_2
  ab=sqrt_sum_2(norm_2(p%a),norm_2(p%b))
  fg=sqrt_sum_2(norm_1(f),norm_1(g))
  zansa_2=norm_2(p%c-p%a*f-p%b*g)/(ab*fg)
end function zansa_2

subroutine Piv_triplet(f,ip,i0,i1)
  ! 多項式列の枢軸選び
  type(triplet_poly),dimension(0:i1),intent(in) :: f
  integer,intent(in) :: i0,i1
  integer,dimension(0:i1),intent(inout) :: ip
  dimension imax(1)
  integer :: i,iw
  imax=maxloc((/(abs(lc(f(ip(i))%c)),i=i0,i1)/))+i0-1
  if (imax(1)==i0) return
  iw=ip(i0)
  ip(i0)=ip(imax(1))
  ip(imax(1))=iw
end subroutine Piv_triplet

subroutine print_triplet(p,n)
  ! 多項式 3 つ組の出力ルーチン
  type(triplet_poly),intent(in) :: p
  integer,optional,intent(in) :: n
```

```
    write(*,*)
    if (present(n)) then
      write(*,*)'p(',n,')(z)='
      call print_polynomial(p%c)
      write(*,*)
      write(*,*)'A(',n,')(z)='
      call print_polynomial(p%a)
      write(*,*)
      write(*,*)'B(',n,')(z)='
      call print_polynomial(p%b)
    else
      write(*,*)'p(z)='
      call print_polynomial(p%c)
      write(*,*)
      write(*,*)'A(z)='
      call print_polynomial(p%a)
      write(*,*)
      write(*,*)'B(z)='
      call print_polynomial(p%b)
    end if
  end subroutine print_triplet

  subroutine poly_to_triplet(f,g,ex_f,ex_g)
    ! 拡張算法の初期設定
    type(polynomial),intent(in) :: f,g
    type(triplet_poly),intent(out) :: ex_f,ex_g
    ex_f%a=1.0_kd ; ex_f%b=0.0_kd ; ex_f%c=f
    ex_g%a=0.0_kd ; ex_g%b=1.0_kd ; ex_g%c=g
  end subroutine poly_to_triplet

end module triplet_system
```

A.3 多項式のアルゴリズムに関するモジュール

```
module polynomial_algorithm
  use polynomial_system

  interface generator
    module procedure generator_poly
  end interface
```



```
  interface Algorithm_A
    module procedure Algorithm_A
    module procedure Algorithm_A_no_base
  end interface

  interface select_prs
    module procedure select_prs
  end interface

contains

  subroutine euclid(f,g,eps,p,t)
    ! Euclid 算法
    type(polynomial),intent(in) :: f,g
    real(kind=kd),intent(in) :: eps
    type(polynomial),dimension(-1:g%deg+1),intent(out) :: p
    integer,intent(out) :: t
    integer :: i
    ! 初期設定 P_{-1}=F, P_{0}=G
    p(-1)=f ; p(0)=g
    do i=0,g%deg
      p(i+1)=p(i-1).mod.p(i)
      call exact_deg(eps,p(i+1)) ! 主係数の零判定
      if (p(i+1)%deg<0) exit
    end do
    t=i
  end subroutine euclid

  subroutine generator_poly(f,g,eps,g1,g2,abnormal,lmax)
    ! 部分生成対 (F,G) の生成
    ! サブルーチン stable_method_sub_prs の補助ルーチン
    ! f: 枢軸多項式
    ! g: 剰余列要素
    ! g1,g2: 隣接する剰余列要素
    type(polynomial),intent(in) :: g
    real(kind=kd),intent(in) :: eps
    type(polynomial),intent(inout) :: f
    type(polynomial),intent(out) :: g1,g2
    logical,intent(out) :: abnormal
```

```
    integer,intent(in),optional :: lmax
    integer :: d,l
    integer,dimension(:),allocatable :: ip
    type(polynomial),dimension(:),allocatable :: w
    allocate(w(0:g%deg))
    abnormal=.false.
    d=f%deg-g%deg
    call step1
    call step2
    call step3
    if (.not.abnormal) call step4
    deallocate(w,ip)

  contains

    subroutine step1
      integer :: j
      w(0)=Shift(g,d-1)
      if (present(lmax)) then
        l=min(g%deg-1,lmax)
      else
        l=g%deg-1
      end if
      j=0
      do while(j<l.and.abs(lc(w(j)))<abs(lc(f)))
        j=j+1
        w(j)=Elim(f,Shift(w(j-1),1))
      end do
      l=j
      w(l+1)=Elim(f,Shift(w(l),1))
    end subroutine step1

    subroutine step2
      integer :: i,j
      allocate(ip(0:l+1))
      ip=(/(i,i=0,l+1)/)
      do i=1,d-1
        call Piv(w,ip,0,l)
        do j=1,l+1
          w(ip(j))=Elim(w(ip(0)),w(ip(j)))
```



```
        end do
        w(ip(0))=Shift(g,d-i-1)
      end do
    end subroutine step2

    subroutine step3
      integer :: j,k
      do k=0,l-1
        call Piv(w,ip,k,l)
        if (abs(lc(w(ip(k))))<=eps) then
          abnormal=.true.
          exit
        else
          do j=k+1,l+1
            w(ip(j))=Elim(w(ip(k)),w(ip(j)))
          end do
        end if
      end do
      if (abs(lc(w(ip(l))))<=eps) abnormal=.true.
    end subroutine step3

    subroutine step4
      real(kind=kd) :: lcp
      lcp=max(abs(lc(w(ip(l)))),abs(lc(w(ip(l+1)))))
      g1=w(ip(l))
      g2=Elim(w(ip(l)),w(ip(l+1)))
      call exact_deg(eps,g2)
      if (abs(lc(w(ip(l))))>=abs(lc(w(ip(l+1))))) then
        f=w(ip(l))
      else
        f=w(ip(l+1))
      end if
    end subroutine step4
  end subroutine generator_poly

  subroutine stable_method_sub_prs(f,g,eps,p,t,abnormal,lmax)
    ! 多項式剰余列の安定な算法（部分終結式算法）
    ! abnormal は 手順 3 の仮定を満足していない場合に True を返す.
    ! lmax は パラメータ l に相当する.
    ! lmax を引数として渡さない場合は，パラメータは自動的に選択される.
```

```
type(polynomial),intent(in) ::f,g
real(kind=kd),intent(in) :: eps
type(polynomial),dimension(-1:g%deg),intent(out) :: p
integer,intent(out) :: t
logical,intent(out) :: abnormal
integer,intent(in),optional :: lmax
integer :: k
type(polynomial),dimension(2) :: w
p(-1)=f ; p(0)=g
if (f%deg>g%deg) then
  w(1)=p(-1); k=0
else
  if (abs(lc(f))>abs(lc(g))) then
    w(1)=p(-1)
  else
    w(1)=p(0)
  end if
  p(1)=Elim(p(-1),p(0))
  call exact_deg(eps,p(1))
  k=1
 end if
 do while(p(k)%deg>0)
  call generator(w(1),p(k),eps,p(k+1),w(2),abnormal,lmax)
  if (abnormal) exit
  if (w(1)%deg<p(k)%deg) then
    k=k+2
  else
    k=k+1
  end if
  p(k)=w(2)
 end do
 t=k
 if (p(t)%deg<0) t=t-1
end subroutine stable_method_sub_prs

subroutine Algorithm_A(f,g,delta,q,l,v,w)
 ! 剰余列の生成 (Givens 回転による算法)
 type(polynomial),intent(in) ::  f,g
 real(kind=kd),intent(in) :: delta
 type(polynomial),dimension(-1:g%deg),intent(out) :: q
```



```
integer,intent(out) :: l
type(polynomial),dimension(g%deg),intent(out) :: v
type(polynomial),dimension(f%deg),intent(out) :: w
type(polynomial) :: s,t
integer :: d,i,k,m,n
m=f%deg ; n=g%deg ; d=m-n

q(-1)=f ; q(0)=g

! 初期設定
call Givens(q(-1),Shift(q(0),d),v(1),s)
t=s
do i=1,d
  call Givens(Shift(q(0),d-i),t,w(i),t)
end do

! 多項式の零判定
if (norm_2(t)>delta*(t%deg+1)) then
  w(d+1)=t
  q(1)=t
else
  l=0
  return
endif

! 反復計算
do k=1,n-1
  call Givens(v(k),Shift(s,1),v(k+1),s)
  t=s
  do i=1,k+d
    call Givens(w(i),t,w(i),t)
  end do

  ! 多項式の零判定
  if (norm_2(t)>delta*(t%deg+1)) then
    w(k+d+1)=t
    q(k+1)=t
  else
    l=k
    return
```

```
    endif
  end do
  l=n
end subroutine Algorithm_A

subroutine Algorithm_A_no_base(f,g,delta,q,l)
  ! 剰余列の生成 (Givens 回転による算法)
  ! 剰余列だけを求める.
  type(polynomial),intent(in) ::  f,g
  real(kind=kd),intent(in) :: delta
  type(polynomial),dimension(-1:g%deg),intent(out) :: q
  integer,intent(out) :: l
  integer :: d,i,k,m,n
  type(polynomial),dimension(:),allocatable :: w
  m=f%deg ; n=g%deg ; d=m-n
  allocate(w(-2:m))

  q(-1)=f ; q(0)=g

  ! 初期設定
  call Givens(q(-1),Shift(q(0),d),w(0),w(-1))
  w(-2)=w(-1)
  do i=1,d
    call Givens(Shift(q(0),d-i),w(-2),w(i),w(-2))
  end do

  ! 多項式の零判定
  if (norm_2(w(-2))>delta*(w(-2)%deg+1)) then
    w(d+1)=w(-2)
    q(1)=w(-2)
  else
    l=0
    deallocate(w)
    return
  endif

  ! 反復計算
  do k=1,n-1
    call Givens(w(0),Shift(w(-1),1),w(0),w(-1))
    w(-2)=w(-1)
```



```
    do i=1,k+d
      call Givens(w(i),w(-2),w(i),w(-2))
    end do

    ! 多項式の零判定
    if (norm_2(w(-2))>delta*(w(-2)%deg+1)) then
      w(k+d+1)=w(-2)
      q(k+1)=w(-2)
    else
      l=k
      deallocate(w)
      return
    endif
  end do
  l=n
  deallocate(w)
end subroutine Algorithm_A_no_base

subroutine select_prs(q,l,delta,t)
  ! 剰余列の選出ルーチン
  type(polynomial),dimension(-1:l),intent(inout) :: q
  integer,intent(in) :: l
  real(kind=kd),intent(in) :: delta
  integer,intent(out) :: t
  integer :: k,r,n
  n=q(0)%deg

  ! 初期化
  k=0 ; r=1
  ! 剰余列の選出
  do while(r<=l)
    k=k+1
    call exact_deg(delta,q(r))
    q(k)=q(r)
    r=n-q(k)%deg+1
  end do
  t=k
end subroutine select_prs

end module polynomial_algorithm
```

A.4 多項式 3 つ組のアルゴリズムに関するモジュール

```
module ex_polynomial_algorithm
  use triplet_system

  interface generator
    module procedure generator_triplet
  end interface

  interface Algorithm_B
    module procedure Algorithm_B
    module procedure Algorithm_B_no_base
  end interface

  interface select_prs
    module procedure select_prs_triplet
  end interface

contains

  subroutine extended_euclid(f,g,eps,p,t)
    ! 拡張 Euclid 算法
    type(polynomial),intent(in) :: f,g
    real(kind=kd),intent(in) :: eps
    type(triplet_poly),dimension(-1:g%deg+1),intent(out) :: p
    integer,intent(out) :: t
    type(polynomial) :: q
    integer :: i
    ! 初期設定 (P_{-1},A_{-1},B_{-1})=(F,1,0)
    !          (P_{0},A_{0},B_{0})=(G,0,1)
    call poly_to_triplet(f,g,p(-1),p(0))
    do i=0,g%deg
      call q_and_mod(p(i-1)%c,p(i)%c,q,p(i+1)%c)
      p(i+1)%a=p(i-1)%a-q*p(i)%a
      p(i+1)%b=p(i-1)%b-q*p(i)%b
      call exact_deg(eps,p(i+1)%c) ! 主係数の零判定
      if (p(i+1)%c%deg<0) exit
    end do
    t=i
```



```
  end subroutine extended_euclid

  subroutine generator_triplet(f,g,eps,g1,g2,abnormal,lmax)
    ! 部分生成対 (F,G) の生成
    ! サブルーチン stable_ex_method_sub_prs の補助ルーチン
    ! f: 枢軸多項式
    ! g: 剰余列要素
    ! g1,g2: 隣接する剰余列要素
    type(triplet_poly),intent(in) :: g
    real(kind=kd),intent(in) :: eps
    type(triplet_poly),intent(inout) :: f
    type(triplet_poly),intent(out) :: g1,g2
    logical,intent(out) :: abnormal
    integer,intent(in),optional :: lmax
    integer :: d,l
    integer,dimension(:),allocatable :: ip
    type(triplet_poly),dimension(:),allocatable :: w
    allocate(w(0:g%c%deg))
    abnormal=.false.
    d=f%c%deg-g%c%deg
    call step1
    call step2
    call step3
    if (.not.abnormal) call step4
    deallocate(w,ip)

  contains

    subroutine step1
      integer :: j
      w(0)=Shift(g,d-1)
      if (present(lmax)) then
        l=min(g%c%deg-1,lmax)
      else
        l=g%c%deg-1
      end if
      j=0
      do while(j<l.and.abs(lc(w(j)%c))<abs(lc(f%c)))
        j=j+1
        w(j)=Elim(f,Shift(w(j-1),1))
```

```
    end do
    l=j
    w(l+1)=Elim(f,Shift(w(l),1))
  end subroutine step1

  subroutine step2
    integer :: i,j
    allocate(ip(0:l+1))
    ip=(/(i,i=0,l+1)/)
    do i=1,d-1
      call Piv(w,ip,0,l)
      do j=1,l+1
        w(ip(j))=Elim(w(ip(0)),w(ip(j)))
      end do
      w(ip(0))=Shift(g,d-i-1)
    end do
  end subroutine step2

  subroutine step3
    integer :: j,k
    do k=0,l-1
      call Piv(w,ip,k,l)
      if (abs(lc(w(ip(k))%c))<=eps) then
        abnormal=.true.
        exit
      else
        do j=k+1,l+1
          w(ip(j))=Elim(w(ip(k)),w(ip(j)))
        end do
      end if
    end do
    if (abs(lc(w(ip(l))%c))<=eps) abnormal=.true.
  end subroutine step3

  subroutine step4
    real(kind=kd) :: lcp
    lcp=max(abs(lc(w(ip(l))%c)),abs(lc(w(ip(l+1))%c)))
    g1=w(ip(l))
    g2=Elim(w(ip(l)),w(ip(l+1)))
    call exact_deg(eps,g2%c)
```



```
    if (abs(lc(w(ip(l))%c))>=abs(lc(w(ip(l+1))%c)) &
      & .or.g2%c%deg<0) then
      f=w(ip(l))
    else
      f=w(ip(l+1))
    end if
  end subroutine step4
end subroutine generator_triplet

subroutine stable_ex_method_sub_prs(f,g,eps,p,t,abnormal,lmax)
  ! 多項式剰余列の安定な拡張算法（部分終結式算法）
  ! abnormal は 手順 3 の仮定を満足していない場合に True を返す.
  ! lmax は パラメータ l に相当する.
  ! lmax を引数として渡さない場合は，パラメータは自動的に選択される.
  type(polynomial),intent(in) ::f,g
  real(kind=kd),intent(in) :: eps
  type(triplet_poly),dimension(-1:g%deg),intent(out) :: p
  integer,intent(out) :: t
  logical,intent(out) :: abnormal
  integer,intent(in),optional :: lmax
  integer :: k
  type(triplet_poly),dimension(2) :: w
  ! 初期設定 (P_{-1},A_{-1},B_{-1})=(F,1,0)
  !          (P_{0},A_{0},B_{0})=(G,0,1)
  call poly_to_triplet(f,g,p(-1),p(0))
  if (f%deg>g%deg) then
    w(1)=p(-1); k=0
  else
    if (abs(lc(f))>abs(lc(g))) then
      w(1)=p(-1)
    else
      w(1)=p(0)
    end if
    p(1)=Elim(p(-1),p(0))
    call exact_deg(eps,p(1)%c)
    k=1
  end if
  do while(p(k)%c%deg>0)
    call generator(w(1),p(k),eps,p(k+1),w(2),abnormal,lmax)
    if (abnormal) exit
```

```
    if (w(1)%c%deg<p(k)%c%deg) then
      k=k+2
    else
      k=k+1
    end if
    p(k)=w(2)
  end do
  t=k
  if (p(t)%c%deg<0) t=t-1
end subroutine stable_ex_method_sub_prs

subroutine Algorithm_B(f,g,delta,q,l,v,w)
  ! 剰余列の生成 (Givens 回転による拡張算法)
  type(polynomial),intent(in) ::  f,g
  real(kind=kd),intent(in) :: delta
  type(triplet_poly),dimension(-1:g%deg),intent(out) :: q
  integer,intent(out) :: l
  type(triplet_poly),dimension(g%deg),intent(out) :: v
  type(triplet_poly),dimension(f%deg),intent(out) :: w
  type(triplet_poly) :: s,t
  integer :: d,i,k,m,n
  m=f%deg ; n=g%deg ; d=m-n

  call poly_to_triplet(f,g,q(-1),q(0))

  ! 初期設定
  call Givens(q(-1),Shift(q(0),d),v(1),s)
  t=s
  do i=1,d
    call Givens(Shift(q(0),d-i),t,w(i),t)
  end do

  ! 多項式の零判定
  if (norm_2(t%c)>delta*(t%c%deg+1)) then
    w(d+1)=t
    q(1)=t
  else
    l=0
    return
  endif
```



```
  ! 反復計算
  do k=1,n-1
    call Givens(v(k),Shift(s,1),v(k+1),s)
    t=s
    do i=1,k+d
      call Givens(w(i),t,w(i),t)
    end do

    ! 多項式の零判定
    if (norm_2(t%c)>delta*(t%c%deg+1)) then
      w(k+d+1)=t
      q(k+1)=t
    else
      l=k
      return
    endif
  end do
  l=n
end subroutine Algorithm_B

subroutine Algorithm_B_no_base(f,g,delta,q,l)
  ! 剰余列の生成 (Givens 回転による拡張算法)
  ! 拡張剰余列だけ求める
  type(polynomial),intent(in) ::  f,g
  real(kind=kd),intent(in) :: delta
  type(triplet_poly),dimension(-1:g%deg),intent(out) :: q
  integer,intent(out) :: l
  integer :: d,i,k,m,n
  type(triplet_poly),dimension(:),allocatable :: w
  m=f%deg ; n=g%deg ; d=m-n
  allocate(w(-2:m))

  call poly_to_triplet(f,g,q(-1),q(0))

  ! w(0) は 基底 V_{k}, w(-1) は S, w(-2) は T に相当する.

  ! 初期設定
  call Givens(q(-1),Shift(q(0),d),w(0),w(-1))
  w(-2)=w(-1)
```

```
do i=1,d
  call Givens(Shift(q(0),d-i),w(-2),w(i),w(-2))
end do

! 多項式の零判定
if (norm_2(w(-2)%c)>delta*(w(-2)%c%deg+1)) then
  w(d+1)=w(-2)
  q(1)=w(-2)
else
  l=0
  deallocate(w)
  return
endif

! 反復計算
do k=1,n-1
  call Givens(w(0),Shift(w(-1),1),w(0),w(-1))
  w(-2)=w(-1)
  do i=1,k+d
    call Givens(w(i),w(-2),w(i),w(-2))
  end do

  ! 多項式の零判定
  if (norm_2(w(-2)%c)>delta*(w(-2)%c%deg+1)) then
    w(k+d+1)=w(-2)
    q(k+1)=w(-2)
  else
    l=k
    deallocate(w)
    return
  endif
end do
l=n
deallocate(w)
end subroutine Algorithm_B_no_base

subroutine select_prs_triplet(q,l,delta,t)
  ! 剰余列の選出ルーチン
  type(triplet_poly),dimension(-1:1),intent(inout) :: q
  integer,intent(in) :: l
```



```
  real(kind=kd),intent(in) :: delta
  integer,intent(out) :: t
  integer :: k,r,n
  n=q(0)%c%deg

  ! 初期化
  k=0 ; r=1
  ! 剰余列の選出
  do while(r<=l)
    k=k+1
    call exact_deg(delta,q(r)%c)
    q(k)=q(r)
    r=n-q(k)%c%deg+1
  end do
  t=k
end subroutine select_prs_triplet
end module ex_polynomial_algorithm
```

A.5 使用例の主プログラム

関数 $f(z)=\frac{e^z}{1+z^3}$ の Maclaurin 展開を $n+\nu$ 項で打ち切った多項式 F と $G=z^{n+1}$ を与えて，アルゴリズム B（Givens 回転による拡張算法）および Padé 近似式の選出原理を使って，$f(z)$ の Padé 近似式およびその誤差推定値を計算した．多項式 F は module polynomial_system で定義した 関数副プログラム exp_p と series_rational を使って生成した．ここでは，$n=20$, $\nu=2$ とした．

```
program main
  use ex_polynomial_algorithm
  type(polynomial) :: f,g
  real(kind=kd) :: eps,gamma,delta,err_estim
  type(triplet_poly),dimension(:),allocatable :: p
  real(kind=kd),dimension(:),allocatable :: err
  integer,parameter :: n=20,nu=2
  integer :: t,i0
  eps=epsilon(eps)
  call sample
  ! 多項式 F の生成
  gamma=sqrt_sum_2(norm_1(f),norm_1(g))
  delta=gamma*eps
```

```
  call generate_pade
  ! Pade 近似式の生成
  call select_pade
  ! Pade 近似式の事後誤差評価による選出
  call select_pade_message
  ! 選出された Pade 近似式の出力
  deallocate(p,err)

contains

  subroutine sample
    type(polynomial) :: q
    q%deg=3
    allocate(q%coef(0:q%deg))
    q%coef=(/1,0,0,1/)
    f=series_rational(exp_p(n+nu),q,n+nu)
    g%deg=n+1
    allocate(g%coef(0:g%deg))
    g%coef(g%deg)=1
    g%coef(0:g%deg-1)=0
  end subroutine sample

  subroutine generate_pade
    integer :: l
    allocate(p(-1:g%deg))
    call Algorithm_B(f,g,delta,p,l)
    call select_prs(p,l,delta,t)
  end subroutine generate_pade

  subroutine select_pade
    real(kind=kd) :: gamma_prime,delta_prime,eps0
    real(kind=kd),dimension(1) :: eps_prime
    integer :: i
    gamma_prime=min(gamma,1.0_kd)
    delta_prime=gamma_prime*eps
    allocate(err(1:t))
    err=(/(norm_2(p(i)%b,nu)/gamma_prime,i=1,t)/)
    eps_prime=minval((/(err(i),i=1,t)/))
    eps0=max(eps,eps_prime(1))
    do i=1,t
```



```
      if (err(i)<=eps0) exit
    end do
    i0=i
    err_estim=err(i0)
    call exact_deg(delta,p(i0)%c) !economization of C
    call exact_deg(delta_prime,p(i0)%a) !economization of A
  end subroutine select_pade

  subroutine select_pade_message
    real(kind=kd) :: a0
    write(*,*)'machine epsilon=',eps
    write(*,*)
    write(*,*)' 選出された有理式'
    write(*,*)
    write(*,*)'C(',i0,')='
    call print_polynomial(p(i0)%c)
    write(*,*)
    write(*,*)'A(',i0,')='
    call print_polynomial(p(i0)%a)
    write(*,*)
    write(*,*)' 誤差推定値=',err_estim
    write(*,*)
    a0=p(i0)%a%coef(0)
    if (a0/=0) then
      write(*,*)' 分母の定数項を 1 に規格化した有理式'
      write(*,*)
      write(*,*)'C(',i0,')='
      call print_polynomial(p(i0)%c/a0)
      write(*,*)
      write(*,*)'A(',i0,')='
      call print_polynomial(p(i0)%a/a0)
    else
      write(*,*)' 分母の定数項が零であるので ABNORMAL'
    end if
  end subroutine select_pade_message

end program
```

実行結果

```
machine epsilon=   2.2204460492503131E-16

選出された有理式

C( 4 )=
(  -0.5091647287659722 )+
(  -0.5091647287659724 ) z^ 1 +
(  -0.2545823643829861 ) z^ 2 +
(  -8.4860788127661957E-02 ) z^ 3 +
(  -2.1215197031915333E-02 ) z^ 4 +
(  -4.2430394063830658E-03 ) z^ 5 +
(  -7.0717323439725699E-04 ) z^ 6 +
(  -1.0102474777108766E-04 ) z^ 7 +
(  -1.2628093471419552E-05 ) z^ 8 +
(  -1.4031214967957784E-06 ) z^ 9 +
(  -1.4031214970894247E-07 ) z^ 10 +
(  -1.2755649910275929E-08 ) z^ 11 +
(  -1.0629708502296609E-09 ) z^ 12 +
(  -8.1767100769347594E-11 ) z^ 13 +
(  -5.8405304283463941E-12 ) z^ 14 +
(  -3.8946623703850491E-13 ) z^ 15 +
(  -2.4238287790634459E-14 ) z^ 16

A( 4 )=
(  -0.5091647287659722 )+
(  -1.3877787807814457E-16 ) z^ 1 +
(  -1.3877787807814457E-17 ) z^ 2 +
(  -0.5091647287659722 ) z^ 3

誤差推定値=   7.0690973745035864E-17

分母の定数項を 1 に規格化した有理式

C( 4 )=
(   1.0000000000000000 )+
(   1.0000000000000004 ) z^ 1 +
(   0.5000000000000001 ) z^ 2 +
(   0.1666666666666665 ) z^ 3 +
(   4.1666666666666324E-02 ) z^ 4 +
(   8.3333333333332638E-03 ) z^ 5 +
(   1.3888888888890331E-03 ) z^ 6 +
(   1.9841269841281906E-04 ) z^ 7 +
(   2.4801587301668363E-05 ) z^ 8 +
(   2.7557319223513934E-06 ) z^ 9 +
(   2.7557319229281152E-07 ) z^ 10 +
(   2.5052108266004456E-08 ) z^ 11 +
(   2.0876757367029545E-09 ) z^ 12 +
(   1.6059066182278756E-10 ) z^ 13 +
(   1.1470807183564520E-11 ) z^ 14 +
(   7.6491205112032737E-13 ) z^ 15 +
(   4.7604019723399034E-14 ) z^ 16

A( 4 )=
(   1.0000000000000000 )+
(   2.7255988138552143E-16 ) z^ 1 +
(   2.7255988138552143E-17 ) z^ 2 +
(   1.0000000000000000 ) z^ 3
```

原則 (5.55) によって，番号 i_0 は 4 に選択され，多項式 C(4)，A(4) はそのときの Padé 近似式の分子と分母である．誤差推定値は式 (5.56) により定められた量である．分母の定数項を 1 に規格化した有理式において，分子は指数関数の Maclaurin 展開を打ち切った多項式，分母は $1+z^3$ にほぼなっている．